Discrete Mathematics and Theory (2120)

Charlie Meyer

MoWeFri 1:00 - 1:50

I realized that MCS and $\forall_x$ are just not helpful after a little while. I began by diligently taking notes on everything, but I quickly realized that they go into much more depth and also just contain information that isn't covered by Elizabeth's course. I will be reading through them but I won't be taking notes on them unless I find something legitimately interesting or helpful. A note for the future readers - focus on Elizabeth's writeups and notes from class/office hours. They are much more helpful - this is just a way to get ahead and really invest yourself in the material.

Logs Writeup

As a summary, you should know that:

- $\log_b(x) = w \equiv b^w = x$
  - Which implies that $b^{\log_b(x)} = x$
- $\log_x(y)$ is approximately the number of digits needed to represent y in base x
  - the exact number is $1 + \lfloor \log_x(y) \rfloor$
- $\log_b(xy) = \log_b(x) + \log_b(y)$ and its related identities:
  - $\log_a(b^n) = n \log_a(b)$
  - $\log_b(\frac{x}{y}) = \log_b(x) - \log_b(y)$
- $\log_b(x) = \frac{\log_a(x)}{\log_a(b)}$

Definitions

The **log base** b **of** x is written as $\log_b(x)$ means "the power to which b must be raised to result in x."

The log base 10 of x is a continuous parallel to the number of digits needed to write x as a decimal (base-10) integer.

The log base 2 of x is a continuous parallel to the number of digits needed to write x as a binary (base-2) integer.

$\ln(x)$ means $\log_e(x)$ where e is 2.71828...l

Properties

How important is the base used in a log?

Example

$\log_3(81) = 4$ because $3^4 = 81; 3^4 = (3^2)^2 = 9^2$ so $\log_9(81) = 2$. But the same reasoning works for other values: if $\log_3(x) = y$, then $3^y = x$, meaning that $(3^2)^{\frac{y}{2}} = x$ meaning $\log_9(x) = \frac{y}{2}$.

Suppose a number used 24 digits to write in base 10. How many digits does it take to write in base 1000? 8. Each cluster of 3 digits in base 10 turns into one digit in base 1000, so $\log_{1000}(x) = \frac{1}{3}\log_{10}(x)$.

Base Identity

$\log_b(x) = \frac{\log_a(x)}{\log_a(b)}$

Base Identity Corollary

$\log_a(b) = \frac{1}{\log_b(a)}$

Logs of Multiples

How much larger is the log of x than the log of $1000x$?

Logs of Multiples Example

Since $\log_1 0(x)$ is the number of digits needed to represent x, and $1000 \times x$ requires exactly 3 more digits, it must be the case that $\log_{10}(1000 \times x) = 3 + \log_{10}(x) = \log_{10} 1000 + \log_{10}(x)$.

Theorem 2

$\log_b(xy) = \log_b(x) + \log_b(y)$

Corollary

$\log_b(x^n) = n \log_b(x)$

$\log_b(\frac{x}{y}) = \log_b(x) - \log_b(y)$

MCS 5.1 - 5.3 and Addendum

MCS 5.1 - 5.3

5.1 - Ordinary Induction

Consider this experiment to understand induction. Imagine a professor brings a bottomless bag of candy to class. She offers to share the candy according to two rules:

1. The student at the beginning gets a candy bar
2. If a student gets a candy bar, then the following student in line also gets a candy bar.

Consider that this sequence has a concise mathematical description: "If a student n gets a candy bar, then student *n+1* gets a candy bar for all nonnegative integers *n*."

5.1.1 - Ordinary Induction The induction principle:

Let P be a predicate on nonnegative integers. If:

- P(0) is true, and
- P(n) implies P(n+1) for all nonnegative integers n, then
- P(m) is trye for all nonnegative integers m.

Formulated as proof, this would be written as:

$$\frac{P(0), \forall_n \in \mathbb{N}.P(N) \to P(n+1)}{\forall_m \in \mathbb{N}.P(m)}$$

5.1.2 - Example Below is the formula for the sum of the nonnegative integers up to n. For all $n \in mathbbN$,

$$1 + 2 + 3 + ... + n = \frac{n(n+1)}{2}$$

To prove by induction, we define a predicate $P(n)$ to be the equation above. Now our job is to prove two things: $P(0)$ is true and that $P(n) \to P(n+1)$ for all $n \in \mathbb{N}$.

First statement follows because of the convention that a sum of zero terms is equal to 0. So $P(0)$ is true.

The second statement is a little more complicated. We assume $P(n)$ to prove $P(n+1)$ which is the equation:

$$1 + 2 + 3 + ... + n + (n+1) = \frac{n(n+1)}{2} + (n+1)$$

Which can be written as:

$$1 + 2 + 3 + ... + n + (n+1) = \frac{(n+1)(n+2)}{2}$$

Thus, if $P(n)$ is true, then so is $P(n+1)$.This argument is valid for every non-negative integer n, so this establishes the fact required by the induction proof. Therefore, the Induction Principle says that the predicate $P(m)$ is true for all nonnegative integers m.

5.1.3 - A template for Induction Proofs

1. **State that the proof uses ordinary induction**. This establishes structure of the proof.
2. **Define an appropriate predicate P(n)**. $P(n)$ is called the *induction hypothesis*. The eventual conclusion will be that $P(n)$ is true for all nonnegative n. A cleary stated induction hypothesis is incredibly important. In most cases, it can be lifted straight from the proposition you are trying to prove.
3. **Prove that P(0) is true**. This is the base case. You might also have to consider multiple base cases depending on the problem.
4. **Prove that P(n) implies P(n+1) for all nonnegative integers n**. This is the inductive step.
5. **Invoke induction**. the induction principle allows you to conclude that $P(n)$ is true for all nonnegative integers n. This is the logical capstone of the proof.

5.2 - Strong Induction

strong induction is useful when a simple proof that the predicate holds for $n+1$ does not follow just from the fact that it holds at n, but from the fact that it holds for other values $\leq n$.

5.2.1 - Rules of Strong Induction Principle of Strong Induction:

Let P be a predicate on nonnegative integers. If:

- P(0) is true, and
- for all $n \in \mathbb{N}, P(0), P(1), ..., P(n)$ together imply $P(n+1)$,

Then $P(m)$ is true for all $m \in \mathbb{N}$

The only change is that a strong induction proof allows you to make more assumption sin the inductive step. In strong induction, you may assume that $P(0), P(1), ...$ and $P(n)$ are *all* true when you go to prove $P(n+1)$. Formulated as a rule, strong induction is written as:

$$\frac{P(0), \forall_n \in \mathbb{N}.(P(0) \wedge P(1) \wedge ... \wedge P(n)) \rightarrow P(n+1)}{\forall_m \in \mathbb{N}.P(m)}$$

Stated more succinctly:

$$\frac{P(0), [\forall_k \leq n \in \mathbb{N}.P(k)] \rightarrow P(n+1)}{\forall_m \in \mathbb{N}.P(m)}$$

5.2.2 - Products of Primes We'll use strong induction to re-prove the fact that every integer greater than 1 is a product of primes.

Proof: We will prove the theorem by strong induction, letting the hypothesis $P(n)$ be "n is a product of primes."

Base Case: $(n = 2)$: $P(2)$ is true because 2 is prime, so it is a length one product of primes by convention.

Inductive Step: Suppose that $n \geq 2$ and that for every number from 2 to n is a product of primes. We must show that $P(n+1)$ holds, namely, that $n+1$ is also a product of primes. We then consider two cases:

1. If $n+1$ is prime, then it is a length one proudct of primes by convention,and so $P(n+1)$ is true.
2. If $n+1$ is not prime, which by definition meanas $n+1 = k \cdot m$ for some integers k, m between 2 and n. Now by the strong induction hypothesis, we know that both k and m are products of primes. By multiplying these products, it follows that $k \cdot m = n+1$ is also a product of primes. Therefore, $p(n+1)$ is true.

So $P(n+1)$ holds for any case, which completes the proof by strong induction that $P(n)$ holds for all $n \geq 2$

5.3 Strong Induction vs. Induction vs. Well Ordering

Any well-ordering proof can be reformatted into an induction proof and the other way around. WOP, induction, and strong induction are ultimately just three ways of presenting the same mathematical reasoning.

Addendum

The idea, programmer's version

Suppose I have a loop that iterates over a list. How do I know what the results will be after the loop is over? To formalize this into a proof strategy, we need:

1. Some starting place, the **base case**.
2. Some loop-like way of getting from one truth to the next. This is the **inductive step**. It generally works as follows:
   - We assume it starts true, this assumption is the **inductive hypothesis**
   - We use that assumption to prove that it will still be true one step later
3. We conclude it must always be true. This is the **principle of induction**.

The idea, formal logic version

The **principle of induction** states that:

$$P(0)$$
$$P(n) \to P(n+1)$$
$$\therefore \forall_x \in \mathbb{N}.P(x)$$

To use it, we first prove $P(0)$ then prove $P(n) \to P(n+1)$ by assuming $P(n)$ and proving $P(n+1)$. Proving $P(0)$ is the **base case** and proving $P(n) \to P(n+1)$ is the **inductive step**. By assuming $P(n)$ we are using the **inductive hypothesis**.

Consider that your base case might include more than just $P(0)$. For example, if you need to prove that all Fibonacci numbers are positive, you might need to define your preducate $P(n)$ to mean "The (n+1)th and (n+2)th Finonacci number are both positive." With this, then your base case becomes P(0) where it is just the 1st and the 2nd are positive.

Example

Consider the code:

```
def babylonain(x):
    y = 1
    for i in range(20):
        y = (y+x/y)/2
    return y
```

How could we prove that the end result is between 1 and x?

Proof: Initially, y is between 1 and x (in particular, it is 1). Each pass through the loop y is updated to be the average of two values: y and $fracxy$. We know one of those (y) is between 1 and x, but is the other? Consider $\frac{x}{y}$. Consider the cases:

1. **Case 1: x < 1**
   - In this case, "y is between 1 and x" means $x \leq y \leq 1$. Because $x \leq y$, $\frac{x}{y} \leq 1$. Because $y \leq 1$, $x \leq \frac{x}{y}$. Thus, $x \leq \frac{x}{y} \leq 1$, meaning it is between 1 and x.
2. **Case 2:** $x \geq 1$
   - In this case, "y is between 1 and x" means "$1 \leq y \leq x$." Because $y \leq x$, $1 \leq \frac{x}{y}$. Because $1 \leq y$, $\frac{x}{y} \leq x$. Thus, $1 \leq \frac{x}{y} \leq x$, meaning it is between 1 and x.

Because $\frac{x}{y}$ is between 1 and x in both cases, it is between them in general. Thus, the loop replaces y with the average of two numbers, both between 1 and x, so it keeps y between 1 and x. Because we start between 1 and x and that doesn't change, the function returns a value that is between 1 and x.

MCS 1.8 and MCS 2 and irrationals and open sets and worksheet and sample solutions

MCS 1.8

A **proof by contradiction** or **indirect proof** shows that if a proposition were false, then some false fact would be true. A general method below is as follows:

In order to prove a proposition P by contradiction:

1. Write, "we use proof by contradiction"
2. Write "Supose P is false"
3. deduce something known to be falce (a logical contradiction)
4. Write "This is a contradiction. Therefore, P must be true."

MCS 2 - Well Ordering Principle

2.1 and 2.2 - Well Ordering Principle and Well Ordering Template

Here is a standard way to organize a proof such that it is easy to see that it is correct to prove that "P(n) is true for all $n \in \mathbb{N}$" using the Well Ordering Principle

- Define the set C of *counterexamples* to P being true. Sepcficially, define

$$C ::= \{n \in \mathbb{N} | \text{not}(P(n)) \text{ is true } \}$$

- Assume for proof by contradiction that C is empty
- By the Well Ordering Principle, there will be a smallest element, n in C.
- Reach a contradiciton somehow - often by showing that *P(n)* is true or by showing that there is another member of set C that is smaller than n.
- Conclude that C must be empty, that is, no counterexamples exist.

Example - Prime Factorization Theorem

Theorem 2.3.1 - Every positive integer greater than one can be factored as a product of primes.

Proof: The proof is by well ordering.

Let C be the set of all integers greater than 1 that cannot be factored as a product of primes. We assume C is not empty and derive a contradiction. If C is not empty, there is at least one element $n \in C$ by well ordeirng. The n can't be prime, because a prime by itself is considered a product of primes and no such products are in C. So n must be a product of two integers a and b where $1 < a, b < n$. Since a and b are smaller than the smallest element in C, we know that $a, b \notin C$. In other words, a can be written as a product of primes $p_1 p_2 ... p_k$ and b as a product of primes $q_1 ... q_l$ Therefore $n = p_1 ... p_k q_1 ... q_l$ can be written as a product of primes, contradicting the claim that $n \in C$. Our assumption that C is not empty must therefore be false.

Proofs of Irrationality (Irrationals)

Non-integral numbers

Theorem 1: $frac23 \notin \mathbb{Z}$.

Proof: Assume that $\frac{2}{3} \in \mathbb{Z}$. That means $\exists_x \in \mathbb{Z}.x = \frac{2}{3}$; i.e. $3x = 2$. By the fundamental theorem of arithmetic, each number has a unique prime factorization, which means that both $3x$ and 2 must have the same prime factors. But 3 is a factor of 3x and not a factor of 2, which is a contradiction. Because assume that $\frac{2}{3} \in \mathbb{Z}$ led to a contradiction, it must be the case that $\frac{2}{3} \notin \mathbb{Z}$.

Irrational Roots

Theorem 2: $\sqrt{2} \notin \mathbb{Q}$.

Proof: Assume that $\sqrt{2} \in \mathbb{Q}$. That means $\exists_{x,y} \in \mathbb{Z}.\frac{x}{y} = \sqrt{2}$ where x and y are relatively prime. Rearranging, we have $x^2 = 2y^2$. By the fundamental theorem of arithmetic, each number has a unique prime factorization, which means that x^2 and $2y^2$ must have the same prime factor. Because x and y are relatively prime, at most one of x and y can have 2 in its prime factorization, we thus processed by cases

1. Case 1: 2 is a factor of x
   - Then x^2 has 2 as a factor with multiplicity ≥ 2. Because 2 is not a factor of y, $2y^2$ has 2 as a factor with multiplicity 1. But $1 < 2$, which is a contradiction.
2. Case 2: 2 is not a factor of x
   - Then x^2 also does not have 2 as a factor, but $2y^2$ does, which is a contradiction.

Because both cases led to a contradiction, assuming $\sqrt{2} \in \mathbb{Q}$ leads to a contradiction in general, which means it must be the case that $\sqrt{2} \notin \mathbb{Q}$.

Irrational Logs

Theorem 3: $\log_2 3 \notin \mathbb{Q}$.

Proof: Assume that $\log_2 3 \in \mathbb{Q}$. That means $\exists_{x,y} \in \mathbb{Z}.\frac{x}{y} = \log_2 3$ where x and y are relatively prime. Rearranging, we have $x = \log_2 3y = \log_2(3^y)$, or $2^x = 3^y$. By the fundamental theorem of arithmetic, each number has a unique prime factorization, which means that both 2^x and 3^y must have the same prime factors. But all of the 2^x's prime factors are 2s and none of 3^y's are, which is a contradiction. Because assuming $\log_2 3 \in \mathbb{Q}$ leads to a contradiction, it must be the case that $\log_2 3 \notin \mathbb{Q}$.■

Open Sets

Lack of Largest and Smallest Values

Theorem 1: there is no smallest rational number larger than 0. That is,

$$\forall_{n_1} \in \mathbb{Q}^+.\exists_{n_2} \in \mathbb{Q}^+.n_2 < n_1$$

Proof: We proceed by contradiction. Asume

$$\neg(\forall_{n_1} \in \mathbb{Q}^+.\exists_{n_2} \in \mathbb{Q}^+.n_2 < n_1)$$

Applying equivalence rules, that is the same as assuming

$$\exists_{n_1} \in \mathbb{Q}^+.\ \nexists_{n_2} \in \mathbb{Q}^+.n_2 < n_1$$

Let n be one such n_1. consider $m = \frac{n}{2}$. Because $n \in \mathbb{Q}^+$, n is positive; because $2 > 1$, we have $0 < \text{m} < \text{n}$. Because $n \in \mathbb{Q}^+$, n is rational, and the division of two rational numbers is rational, so m must be rational, and because $0 < m$ we have $m \in \mathbb{Q}^+$. But that contradicts our assumption that $\nexists_{n_2} \in \mathbb{Q}^+.n_2 < n_1$. Because our assumption led to a contradiction, our assumption must be false; that is

$$\forall_{n_1} \in \mathbb{Q}^+.\exists_{n_2} \in \mathbb{Q}^+.n_2 < n_1$$

must be true. ■

There are more corrolaries to this theorem but I'm not taking notes on them. read the in your own time.

Open Sets

Definition 1: A non-empty set is said to be **open** if its members are ordered but it has no largest or smallest element.

Theorem 3: For any $q_1, q_2 \in \mathbb{Q}.q_1 < q_2$, the set defined as $\{q | q \in \mathbb{Q} \wedge q_1 < q < q_2\}$ is an open set.

Proof: Let $A(q_1, q_2) = \{q | q \in \mathbb{Q} \wedge q_1 < q < q_2\}$. The smallest element of $A(q_1, q_2)$ would be then the smallest rational number larger than q_1 which, per corollary 1, does not exist. So $A(q_1, q_2)$ has no smallest element. By a similar logic and corollary 2, $A(q_1, q_2)$ has no largest element. ■.

4.3 Functions; 4.4 Relations

4.3.1 - Domains and Images

a *function* assigns an element of one set, called the *domain*, to an element of another set, called the *codomain*. The notation:

$$f : A \to B$$

shows that f is a function with a domain A, and codomain B. The familiar notation $f(a) = b$ indicates that f assigns the element $b \in B$ to a. Here b would be called the *value* of f at a.

A function with finite domain could be specified by a table. For example, $f_4(P, Q)$ where P and Q are propositional variables specified by the table:

P	Q	f_4(P, Q)
T	T	T
T	F	F
F	T	T
F	F	T

Note that f_4 could also have been described by a formula:

$$f_4(P, Q) ::= [\ \text{P implies Q}\ ]$$

A function can also defined a procedure for computing its value at any element in its domain.

4.3.2 - Composition of Functions

Composing functions f and g means that first f is appleid to some argument, x, to produce *f(x)*, and then g is applied to that result to produce *g(f(x))*.

Definition 4.3.1: for functions $f : A \to B$ and $g : B \to C$, the *composition*, $g \circ f$, of g with f is to be the function from A to C defined by the rule:

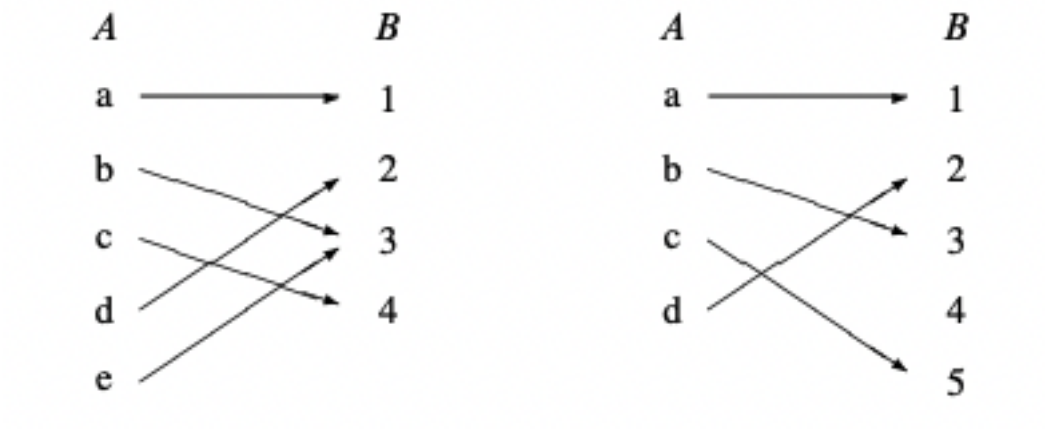
$$(g \circ f)(x) = g(f(x))$$

4.4: Binary Relations

Binary relations define relations between two objects. The relation "less than", for example, on the real numbers relates every real number a to a real number b, precisely when a is less than b.

Definition 4.4.1: A *binary relation*, R, consists of a set, A, called the *domain* of R, a set, B called on the *codomain* of R, and a subset of $A \times B$, called the *graph of* R.

A relation whose domain is A and codomain is B is said to be "between A and B" or "from A to B". We write $R : A \to B$ to indicated that R is a relation from A to B. There are some examples below:



Definition 4.4.2: A binary relation R is:

- A function when it has the [$\leq$ 1 arrow out] property - every point in the domain column has *at most one arrow coming out of it.*
- *surjective* when it has the [$\geq$ 1 arrow in] property - that is, every pint in the righthand, the codomain, has at least one arrow pointing to it.
- *total* when it has the [$\geq$ 1 arrows out] property.
- *injective* when it has the [$\leq$ 1 arrow in] property.
- bijective when it has both the [$=$ 1 arrow out] and the [$=$ 1 arrow in] properties.

Using these definitions, we can see that the diagram of the relation on the left has [$=$ 1 out] and [$\geq$ 1 in]. This means that it is a surjective function. The diagram on the right has the [$=$ 1 out] and [$\leq$ 1 in] properties, which means it is an injective function.

4.4.2 - Relational Images

The idea of an image of a set under a function extends directly to relations. **Definition 4.4.4**: the *image* of a set Y under a relation R, written $R(Y)$ is the set of elements of the codomain, B of R that ar related to some element in Y. In terms of the relation diagram, $R(Y)$ is the set of points with an arrow coming in that starts from some point in Y.

Inverse Relations and Images

Definition 4.4.5: The *inverse*, R^{-1} of a relation $R : A \to B$ is the relation from B to A defined by the rule:

$$bR^{-1}a \text{ iff a R b}$$

in other words, R^{-1} is the relation you get by reversing the direction of the arrows in diagram R.

Definition 4.4.6: The image of a set under the relation R^{-1} is called the *inverse image* of the set. That is, the inverse image of a set X under the relation R is defined to be $R^{-1}(X)$.

4.4.1 - Relation Diagrams

The diagram for a binary relation, R, has points corresponding to the elements of the domain appearing in one column. ALl elements of the codomain appear in another column (usually on the right).

$\forall_x 11.4 - 11.7$

$\forall_x 11.4 - 11.7$

This was extra notes tacked on that wasn't in the calendar initially.

11.4 - Entailment and Validity

The sentences $A_1, A_2, ..., A_n$ entail the sentence L if there is no valuation of the sentence which makes all of $A_1, A_2, ..., A_n$ true and L false.

Consider this with a truth table. Let us check whether $\neg L \rightarrow (J \vee L)$ and $\neg L$ entail J. We simply need to check whether there is any valuation which makes both $\neg L \rightarrow (J \vee L)$ and $\neg L$ true whilst making J false:

J	L	$\neg L \rightarrow (J \vee L)$	$\neg L$	J
T	T	F T **T** T T T	**F** T	**T**
T	F	T F **T** T T F	**T** F	**T**
F	T	F T **T** F T T	**F** T	**F**
F	F	T F **F** F F F	**T** F	**F**

As we can see, only the row on which $\neg L \rightarrow (J \vee L)$ and $\neg L$ are true is the *second* row, and that is a row in which J is also true. Therefore, $\neg L \rightarrow (J \vee L)$ and $\neg L$ entail J.

11.5 - The limits of these tests

Consider the argument "Jan is neither bald nor not-bald." We could express this as $\neg J \wedge \neg\neg J$, and this is a contradiction. You notice that TFL is just not good sometimes! TFL can only handle truth-functional connectives.

11.6 - The Double Turnstile

The double turnstile $\vDash$ is used in this expression:

$$A_1, A_2, ..., A_n \vDash L$$

This shows that the TFL sentences $A_1, A_2, ..., A_n$ together entail L. The $\vDash$ is the *double turnstile*. A way to understand this is that the expression $P, P \rightarrow Q \vDash Q$ is just an abbreviation for the english sentence "The TFL sentences 'P' and '$P \rightarrow Q$' entail 'Q.'

11.7 $\vDash$ versus $\rightarrow$

$A \vDash L$ if and only if there is no valuation of the sentence letters that makes A true and L false. $A \rightarrow L$ is a tautology if and only if there is no valuation of the sentence letters that makes $A \rightarrow L$ false. Thus we can conclude that $A \rightarrow L$ is a tautology iff $A \vDash B$.

Definitions

- $\rightarrow$ is a sentential connective of TFL
- $\vDash$ is a symbol of augmented English

$\forall_x 21.4 - 24$ and MCS 3.6 and skim $\forall_x 25, 26$ and bus example, and practice exercises in $\forall_x$ 22, 23 and Spring 2020 Class Examples

$\forall_x 21.4 - 24$

Quantifiers

The symbol $\forall$ is the **universal quantifier**. So, for example, we could symbolize the sentence "Everyone is happy" with the expression $\forall_x H(x)$. The expression $\forall_x$ represents that you can pick anyone and put them in x, and the subsequent H(x) means, of that thing you picked, it is happy.

The **existential quantifier**, $\exists$, also requires a variable. We can represent the sentence "Someone is angry" as $\exists_x A(x)$ - "there is something, x ,such that x is angry."

More examples of these quantifiers include:

- No one is angry: $\neg\exists_x A(x)$
- There is someone who is not happy: $\exists_x \neg H(x)$
- Not everyone is happy: $\neg\forall_x H(x)$

Domains

Domain is a collection of things that we are talking about. Quantifiers *range over* the domain. Additionally, in FOL, the domain must include at least one thing.

- A domain must have *at least* one member. A name must pick out *exactly* one member of the domain, but a member of the domain may be picked out by one name, many names, or none at all.

Non-referring terms Each name must pick out exactly one member of the domain. A name cannot refer to more than one thing; it is a *singular* term. If you are talking about something that isn't in the domain, any sentence/expression is *meaningless* if it talks about a non-existent thing.

Common quantifier phrases

Take these four example sentences:

1. Every coin in my pocket is a quarter
2. Some coin on the table is a dime
3. Not all the coins on the table are dimes
4. None of the coins in my pocket are dimes.

And take these four statements under the domain "all coins:"

1. P(x) _____x is in my pocket
2. T(x) _____x is on the table
3. Q(x) _____x is a quarter
4. D(x) _____x is a dime

A sentence can be symbolized as $\forall_x(F(x) \rightarrow G(x))$ if it can eb paraphrased in English as "every F is G." We can symbolize sentence (1) as $\forall_x(P(x) \rightarrow (Q(x))$. this is because it is paraphrased as "for any coin, *if* that coin is in my pocket *then* it is a quarter." We can symbolize sentence (2) as $\exists_x(T(x) \rightarrow D(x))$

A sentence can be symbolized as $\exists_x(F(x) \wedge G(x))$ if it can be paraphrased in English as 'some F is G.' Sentence (3) can be paraphrased as "it is not the case that every coin on the table is a dime," so it can be written $\neg\forall_x(T(x) \rightarrow D(x))$ or $\exists_x(T(x) \wedge \neg D(x))$ Finally, Sentence (4) can be paraphrased as "it is not the case that there is some dime in my pocket" and symbolized as $\neg\exists_x(P(x) \wedge D(X))$. Note how this can also be written as $\forall_x(P(x) \rightarrow \neg D(x))$.

Empty Predicates

A predicate that applies to nothing in the domain is called an **empty predicate**. Suppose we wanted to symbolize the two sentences in the domain of animals

1. every monkey knows sign language
2. Some monkey knows sign language

- M(x): _____x is a monkey
- S(x): _____x knows sign language

Sentence (1) can be symbolized as $\forall_x(M(x) \rightarrow S(x))$ and sentence (2) as $\exists_x(M(x) \wedge (S(x))$. Consider the scenario where the domain contains no monkeys. it is possible for sentence (1) to be true while sentence (2) is false.

This idea can be boiled down into this definition: When F is an empty predicate, a sentence $\forall_x(F(x) \rightarrow ...)$ will be vacuously true.

Picking a Domain

Picking domains changes the way that you write sentences in FOL. For example, the sentence "every rose has a thorn" in the domain of "all roses by *only* roses" can be expressed as $\forall_x T(x)$ (where T(x) is **_x has a thorn). However, if we were to expand the domain to "people and plants," then the sentence "every rose has a thorn" is symbolized by** $\forall_x(R(x) \rightarrow T(x))$**, where T(x) is** _x has a thorn and R(x) is _____x is a rose.

Paraphrase and Scope

For the two sentences over the domain "people:"

1. If Kim is a bassist, then she is a rockstar
2. If a person is a bassist, then she is a rockstar.

- B(x): _____x is a bassist
- R(x): _____x is a rock star

Sentence (1) is expressed as $B(k) \to R(k)$ whereas sentence (2) is expressed as $\forall_x(B(x) \to R(x))$

Now what if we wanted to express the sentences:

1. If everyone is a bassist, then Lars is a bassist
2. If everyone is such that, if they are a bassist, then Lars is a bassist

We can express sentence (1) as $\forall_x B(x) \to B(l)$ where l represents Lars. We can also express sentence (2) as $\forall_x(B(x) \to B(l))$. Notice the difference in parenthesis/scope.

Many-placed predicates

Other predicates conern the *relation* between two things. These are **two-place** predicates. So, in order to express sentences ike _____ loves _____, then we need to keep track of the gaps.

In the domain of "people," let i be *Irme* and k be *Karl*. We can express the sentence "Karl loves Irme" as L(k, i). Sentences like "Karl loves Irme, but not vice versa" can be written as $L(k, i) \land \neg L(i, k)$. Make sure to pay attention to the order of the places!

Order of Quantifiers

Consider:

1. For every person x, there is some person that x loves
2. There is some particular person whom every person loves.

We can symbolize (1) as $\forall_x \exists_y L(x, y)$. Sentence (2) can be symbolized as $\exists_y \forall_x L(x, y)$. The point is, make sure to keep track of order. A simple example to emphasize this is the logical fallacy "every dog has its day $\therefore$ there is a day for all the dogs." Nope! That's cringe :open_mouth:

More Symbolization Examples

Over the domain "people and dogs," and

- D(x): _____x is a dog
- F(x, y): _**x is a friend of** _y
- O(x, y): _**x owns** _y
- g: Geraldo

Let's try to symbolize:

1. Geraldo is a dog owner
2. Someone is a dog owner
3. All of Geraldo's friends are dog owners
4. Every dog owner is a friend of a dog owner

- Sentence (1) can be paraphrased as "There is a dog that Geraldo owns": $\exists_x(D(x) \land O(g, x))$
- Sentence (2) can be paraphrased as "There is some y such that Y is a dog owner": $\exists_y$(y is a dog owner) -> $\exists_y \exists_x(D(x) \land O(y, x))$. Notice how we basically just put sentence (1) inside of of the "y is a dog owner" part! :cool:
- Sentence (3) can be paraphrased as "Everyone who is a friend of Geraldo is a dog owner": $\forall_x[F(x, g) \to$ x is a dog owner] -> $\forall_x[F(x, g) \to \exists_z(D(z) \to O(x, z))]$
- Sentence (4) can be written as "For any x that is a dog owner, there is a dog owner who x is a friend of." $\forall_x$[x is a dog owner $\to$ $\exists_y$(y is a dog owner $\land F(x, y)$)] -> $\forall_x[\exists_z(D(z) \land O(x, z)) \to \exists_y(\exists_z(D(z) \land O(y, z)) \land F(x, y))]$.

Notice how we used previous statements and built off of them to create more complex sentences.

Suppressed Quantifiers

Go read that yourself i'm not paraphrasing 17th century english writing.

Identity

Let's say you have a domain of "people" and have the statement O(x, y) _**x owes money to** _y. If you were to say "Pavel owes money to everyone," we write this as $\forall_x O(p, x)$ but this implies that Pavel owes money to himself. We can make reasonable assumptions from living on Earth and understanding the horrifically difficult english language that we meant something like

1. Pavel owes money to everyone else
2. Pavel owes money to everyone other than Pavel
3. Pavel owes money to everyone except Pavel himself

but that wasn't encoded in the original FOL statement. So how do we express these things??

Adding Identity

The symbol '=' is a two-place predicate: x=y **_x is identical to _y**. So if we were to build upon the examples from the identity section, we can express the sentences above as $\forall_x(\neg x = p \rightarrow O(p, x))$. This is understood as "for all x, if x is not Pavel, then x is owed money by Pavel."

There are at least...

Consider the sentences

1. There is at least one apple
2. There are at least two apples
3. There are at least three apples

- A(x) _____x is an apple

We can symbolize (1) as $\exists_x A(x)$. However, sentence two has more complexity. Not only does it have to state that there are at least two apples, but it also has to encode that there are two *unique* apples. It thus can be symbolized as $\exists_x \exists_y((A(x) \wedge A(y)) \wedge \neg x = y)$ Continuing with this trend, if we wanted to specify that there are at least *three unique* apples, then we'd have to say $\exists_x \exists_y \exists_z[((A(x) \wedge A(y)) \wedge A(z)) \wedge ((\neg x = y \wedge \neg y = z) \wedge \neg x = z)]$.

There are at most...

Now consider

1. There is at most one apple
2. There are at most two apples

Sentence (1) can be paraphrased also as "it is not the case that there are at least two apples" or that "if you picked up an apple and then picked up another apple, you picked up the same apple." So you can write it as $\neg\exists_x \exists_y[(A(x) \wedge A(y)) \wedge \neg x = y]$ or $\forall_x \forall_y[(A(x) \wedge A(y)) \rightarrow x = y]$. Same thing with sentence (2), just add the appropriate amount of variables to consider the possibility of two unique apples at most. I'm not gonna write it out.

There are exactly...

Consider

1. There is exactly one apple
2. There are exactly two apples

(1) can be understood as "there is at least one apple *and* at most one apple so thus can be written as $\exists_x A(x) \wedge \forall_x \forall_y[(A(x) \wedge A(y)) \rightarrow x = y]$, but you can also understand sentence (1) as just"There is a thing x which is an apple, and everything which is an apple is just x itself" which can be written as $\exists_x[A(x) \wedge \forall_y(A(y) \rightarrow x = y)]$

(2) can be paraphrased as "There are at least two different apples, and every apple is one of those two apples." This then can be written as $\exists_x \exists_y[((A(x) \wedge A(y)) \wedge \neg x = y) \wedge \forall_z(A(z) \rightarrow (x = z \vee y = z))]$

Finally, consider the very bland statement "There are exactly two things" or "there are exactly two objects." Words like "thing" or "object" apply trivially to everything. So we can thus represent them as $\exists_x \exists_y[\neg x = y \wedge \forall_z(x = z \vee y = z)]$.

MCS 3.6 - Predicate Formulas

Quantifiers

Here are some ways to express "always true" and "sometimes true"

Always True:

Statement	ReWritten
For all $x \in D, P(x)$ is true	For all $x \in \mathbb{R}, x^2 \geq 0$
P(x) is true for every x in the set D	$x^2 \geq 0$ for every $x \in \mathbb{R}$

Sometimes True

Statement	ReWritten
There is an $x \in D$ such that P(x) is true	There is an $x \in \mathbb{R}$ such that $5x^2 - 7 = 0$
P(x) is true for some x in the set D.	$5x^2 - 7 = 0$ for some $x \in \mathbb{R}$

All of these sentences "quantify" how often the predicate is true. Specifically, an asserting that a predicate is always true is called **universal quantification** and an assertion that a predicate is sometimes true is called **existential quantification**.

Mixing Quantifiers

Be careful of the order in which you present quantifiers, for it can change the meaning of a sentence.

Variables over one domain

When all the variables in a formula are taken from the same domain, you don't have to mention that every time. For example, instead of $\forall_x \in D \exists_y \in D.Q(x, y)$ we can just write $\forall_x \exists_y .Q(x, y)$.

Negating Quantifiers

Two examples of negation of quantifiers:

$\neg(\forall_x.P(x))$ is equivalent to $\exists_x \neg(P(x))$. An example of this in english is:

1. Not every likes ice cream.
2. There is someone who does not like ice cream.

$\neg(\exists_x.P(x))$ is equivalent to $\forall_x \neg(P(x))$. AN example of this in english is:

1. There is no one who likes being mocked.
2. Everyone dislikes being mocked.

Validity of Predicate Formulas

For a predicate formula to be valid it must be true no matter what the domain of discourse may be.

Skim $\forall_x$Chapters 25 & 26

The notes are only on the very important things of this chapter since I only skimmed it.

There are 6 kinds of symbols in FOL

1. Predicates - *A, B, C, D, . . . , Z* or with subscripts.
2. Names - *a, b, c, . . . , z* or with subscripts.
3. Variables - *s, t, u, v, w, x, y, z* or with subscripts
4. Connectives - $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
5. Brackets - (,)
6. Quantifiers - $\forall, \exists$

Bus Example

Introduction

Question - What does the statement "everyone can fit in a bus" mean? Answer: It depends on **context**. So, what is context? Context is the change in the likely meaning of an ambiguous word or situation. How do we know what a given context means? Some of it is intrinsic ("this part has to mean x or that part won't make sense") whereas other parts are mostly *cultural*, i.e. we assume the statement "everyone can fit into a car to mean. . . "

Everyone can fit in a	Suggests (to me)
car	Partition: several cars, more car seats than people
costume	Any can fit each: costumes are baggy enough for any build
auditorium	Any can fit in each seat: each auditorium has more seats than people

Partition

Everyone can fit in a bus by dividing people between buses

$\exists$: A partition.

Let B be the set of buses, P be the set of people and c: $B \rightarrow \mathbb{N}$ be a function giving the capacity of a bus. Then this case is:

$$\exists_f : B \rightarrow \mathcal{P}(P).(\forall_b \in B.|f(b)| \leq c(b)) \wedge (\forall_p \in P.\exists_b \in B.p \in f(b))$$

That is,

- "there's some mapping from buses to sets of people that both"
  - "$\exists_f : B \to \mathcal{P}(P)$."
- "the number of people mapped to each bus is within the bus' capacity"
  - "$(\forall_b \in B.|f(b)| \leq c(b))$"
- "and every person s in the set mapped to by some bus"
  - "$(\forall_p \in P.\exists_b \in B.p \in f(b))$"

Any can fit in the set (set fits)

Everyone can fit in a bus, so we only need one bus

∀ buses, set fits.

Let B be the set of buses, P be the set of people, and *f(x, y)* be a predicate asserting that x can fit in y. Then this case is:

$$\exists_b \in B.f(P, b)$$

Any can fit in each

everyone can fit in a bus, even the largest person in the world.

∀ person∀ bus, person fits in bus

This is formalized in mathematics by saying the "fit in a bus" predicate applies to any person we happen to pick. Let B be the set of buses, P be the set of people, and *f(x, y)* be a predicate asserting that x can fit in y. Then:

$$\forall_p \in P, b \in B.f(p, b)$$

One can fit in the set

Everyone can fit in a bus, so we only need one of the big buses.

∃ bus, set fits

When we say "a bus" we mean one bus. In math, we distinguish these two ideas: $\forall_x$ means "no matter which x we pick" and $\exists_x$ means "it is possible to pick the right x." Let B be the set of buses, P be the set of people, and *f(x, y)* be a predicate asserting x can fit in y. Then:

$$\exists_b \in B.f(P, b)$$

One can fits each

Everyone can fit in a bus; even the largest person in the world can fit on a big bus

∃ bus ∀ person perosn fits in a bus

Let B be the set of buses, P be the set of people, and *f(x, y)* be a predicate asserting x can fit in y. Then the case is:

$$\exists_b \in B.\forall_p \in P.f(p, b)$$

Each has one that can fit it

everyone can fit in a bus; there are buses with high doors and ceilings for tall people, buses with wide doors and aisles for wide people, and so on.

∀ person ∃ bus, person fits in bus

Let B be the set of buses, P be the set of people, and *f(x, y)* be a predicate asserting x can fit in y. Then the case is:

$$\exists_p \in P.\exists_b \in B.f(p, b)$$

Practice Exercises in $\forall_x$ Chap. 22 & 23

Not doing them since it's more important to do practice she gives us first. May go back to this later, but it could also be in the "practice" section.

Spring 2020 Class Examples (English to Quantifiers)

domain:	people
H(x):	x is happy
C(x):	x is in this class
A(x, y):	x appreciates y
t :	The Teacher

- Everyone is happy:
  - $\forall_x.H(x)$
- Everyone in this class is happy
  - $\forall_x.C(x) \rightarrow H(x)$
- Someone is happy:
  - $\exists_x.H(x)$
- Someone in this class is happy
  - $\exists_x.C(x) \wedge H(x)$
- Everyone is not happy:
  - $\neg\forall_x.H(x)$
- Someone is unhappy:
  - $\neg\exists_x.H(x)$
- Only one person is happy (there are multiple ways of saying this):
  - "Someone is happy, and everyone other than them is unhappy"
    - $\exists_x.H(x) \wedge \forall_y.((x \neq y) \rightarrow \neg H(y))$
  - "If two people are happy, they are the same person."
    - $\exists_x.\exists_y.(H(x) \wedge H(y)) \rightarrow (x = y) \wedge (\exists_z.H(z)$
    - Note that this both includes "if two people are happy, then they are the same person" and "there is at least one person who is happy."
  - "If two people are distinct, at least one is unhappy."
    - $\exists_{x,y}.(x \neq y) \rightarrow (\neg H(x) \vee \neg H(y)) \wedge \exists_z.H(z)$
    - Note that this both includes the "if two people are distinct, then at least one is unhappy" and "at least one person is happy."
- Only one person in this class is happy (there are multiple ways of saying this):
  - "Someone in the class is happy, and everyone in the class other than them is unhappy."
    - $\exists_x.(C(x) \wedge H(x)) \wedge (\forall_y.(C(y) \wedge (x \neq y)) \rightarrow \neg H(y))$
  - "If two people are both in the class and both happy, they are the same person"
    - $\exists_{x,y}.(C(x) \wedge C(y) \wedge H(x) \wedge H(y)) \rightarrow (x = y)$
- Everyone appreciates someone (for everyone, there is someone they appreciate)
  - $\forall_x.\exists_y.A(x, y)$
- Everyone appreciate someone else (For everyone, there is someone (not them) that they appreciate)
  - $\forall_x.\exists_y.(A(x, y) \wedge x \neq y)$
- Everyone appreciates someone who appreciates them (due to ambiguity, there are multiple interpretations):
  - "For everyone, there is someone that they appreciate and appreciates them"
    - $\forall_x.\exists_y.(A(x, y) \wedge A(y, x))$
  - "For everyone, they appreciate everyone who appreciates them"
    - $\forall_x.\forall_y.A(x, y) \rightarrow A(y, x)$ or $\forall_{x,y}.A(x, y) \rightarrow A(y, x)$ or $\forall_{x,y}.A(x, y) \leftrightarrow A(y, x)$
- Everyone appreciates someone else who appreciates them
  - For everyone, there is someone (not them) that they appreciate and appreciates them:
    - $\forall_x.\exists_y.(x \neq y) \wedge A(x, y) \wedge A(y, x)$
    - There's some ambuguity there, so We should write it more formally as $\forall_{x,y}.(x \neq y) \rightarrow (A(x, y) \leftrightarrow A(y, x))$
- Everyone appreciates a person who appreciates them
  - If "a person" means "everyone," we have:
    - $\forall_x.\forall_y.A(x, y) \rightarrow A(y, x)$
  - If "A person" means "someone"
    - $\forall_x.\exists_y.A(x, y) \wedge A(y, x)$
- Everyone in this class appreciates someone in this class
  - For everyone if they are in this class then there is someone who is in this class and they appreciate them:
    - $\forall_x.C(x) \rightarrow (\exists_y.C(y) \wedge A(x, y))$
    - Can be rewritten as $\forall_x.\exists_y.C(x) \rightarrow (C(y) \wedge A(x, y))$
- There's someone in this class that everyone in the class appreciates
  - $\exists_x.C(x) \wedge (\forall_y.C(y) \rightarrow A(x, y))$
- Those in this class only appreciate people in this class
  - For anyone, if you are in the class then for anyone, if they are not in the class you don't appreciate them

    * $\forall_x.C(x) \to (\forall_y.\neg C(y) \to \neg A(x,y))$
  - For any two people, if one appreciates the other and one is in the class, so is the other
    * $\forall_{x,y}.((A(x,y) \wedge C(x)) \to C(y)$
    * Can also be written as $\forall_{x,y}.A(x,y) \to (C(x) \to C(y))$
- The teacher only appreciates those who appreciate someone in class
  - For anyone, if the teacher appreciates them then there is someone in the class they appreciate
    * $\forall_x.A(t,x) \to (\exists_y.C(y) \wedge A(x,y))$
  - There's someone in the class such that anyone the teacher appreciates appreciates that person.
    * $\exists_x.C(x).\forall_y.A(t,y) \to A(x,y)$

Predicates: MCS 1.2 and $\forall_x$ 21.0 - 21.3 and skim MCS 4.1.0

Predicates

A **predicate** is a proposition whose truth depends on the value of one or more variables. So "n is a perfect square" is a predicate, as you don't know whether it's true or false until you know the value of n.

We use special notation to mark a predicate, very similar to function notation in math: $P(n)$ ::= "n is a perfect square." Note that this is a *boolean*! It returns either true (n is a perfect square) or false (n is not a perfect square).

21.1 - Decomposing Sentences

Consider the sentence "Will is a logician. All logicians wear funny hats. ∴ will wears a funny hat." It is not sufficient to break down this sentence into three points in TFL, as the sentence "all logicians wear funny hats" is not a proposition that can be encoded in one letter. Thus, we need to use *first-order logic*, or *FOL* to achieve this. Here's the basic idea of *FOL*:

- **Names**: we indicate these with lowercase letters. For instance, 'b' might stand for Bertie.
- **Predicates**: predicates are expressions like '____ is a dog' or '____ is a logician.' They are sentences that are not complete on their own but need a name to be completed. We use uppercase letters to represent predicates.
  - The expression D(b) will be a sentence that symbolizes "Bertie is a dog."
- **Quantifiers**: For instance ∃ represents "There is at least one…", so we could read out $\exists_x D(x)$ as 'there is at least one thing, x, such that x is a dog.'

21.2 - Names

A *singular term* is a word/phrase referring to a *specific* person or place or thing. *Proper names* pick out individuals. In FOL, names are lowercase letters 'a' through 'r.' We can also add subscripts to them like $a, b, c, ..., r, a_1, f_{32}, j_{390}$.

21.2 - Predicates

The simplest predicates are just properties of individuals (names). In FOL, predicates are capital letters A through Z, with or without subscripts. We might write them as:

- $A(x)$: _____ $_x$ is angry.
- $H(x)$: _____ $_x$ is happy.

You may start to see how we can symbolize English sentences with these notations. For example:

1. "Elsa is angry"
   - $A(e)$
2. "Gregor and Marybeth are angry"
   - $A(g) \wedge A(e)$
3. "If elsa is angry, then so are Gregor and Marybeth"
   - $A(e) \to (A(g) \wedge A(m))$

MCS 4.1.0

Sets

The conventional set is written C = {red, green, blue} or something else. This works fine for small finite sets. Others can be indicated by how to generate a list of them: $D ::= \{1, 2, 4, 8, 16, ...\}$. Also recall that the order of the elements is not significant.

Popular Sets

Symbol	Set	elements
∅	the empty set	none

Symbol	Set	elements
$\mathbb{N}$	nonnegative integers	0, 1, 2, 3...
$\mathbb{Z}$	integers	...3, 2, 1, 0, 1, 2, 3...
$\mathbb{Q}$	rational numbers	$\frac{1}{2}$ 16, etc
$\mathbb{R}$	real numbers	π, e, $\sqrt{2}$
$\mathbb{C}$	complex numbers	i, $\frac{19}{2}$, etc.

Comparing and Combining Sets

$S \subseteq T$ indicates that the set S is a *subset* of set T, which means that every element of S is also an element of T. Think of it as $\leq$. This is stuff you already learned elsewhere so I won't take notes on it again.

Equivalence Proofs and Direct Proof: MCS 3.4.2 and $\forall x$ 14-15 and equivalences and example proof and proof techniques and direct proof

MCS 3.4.2 - Proving Equivalences

A proposition with n variables has 2^n lines, so adding more and more propositions can become more and more tedious. An alternative approach is to use algebra to prove equivalence. To do this, try to remove all things except *not*, *and*, and *or*. Trick such as $A \to B$ can be turned into not(A) or (B). Below is a list of equivalence axioms.

- A and B $\leftrightarrow$ B and A (commutativity of *and*)
- (A and B) and C $\leftrightarrow$ A and (B and C) (associativity of *and*)
- $\top$ and A $\leftrightarrow$ A (identity for *and*)
- $\bot$ and A $\leftrightarrow$ $\bot$ (zero for *and*)
- A and A $\leftrightarrow$ A (impotence for *and*)
- A and $\bar{A} \leftrightarrow \bot$ (contradiction for *and*)
- not($\bar{A}$) $\leftrightarrow$ A (double negation)
- A or $\bar{A} \leftrightarrow \top$
- not(A and B) $\leftrightarrow$ $\bar{A}$ or $\bar{B}$ (DeMorgan for *and*)
- not(A or B) $\leftrightarrow$ $\bar{A}$ and $\bar{B}$ (DeMorgan for *or*)

Example

Here's an example of converting any formula into a normal form. Take, for example, not((A and B) or (A and C)).

1. Start by applying DeMorgan's Law:
    1. not(A and B) and not(A and C)
2. Now Apply DeMorgan's Law again:
    1. ($\bar{A}$ or $\bar{B}$) and ($\bar{A}$ or $\bar{C}$)
3. Now apply the distributivity of *and* over or by distributing ($\bar{A}$ or $\bar{B}$) to get:
    1. (($\bar{A}$ and $\bar{A}$) or ($\bar{B}$ and $\bar{A}$)) or (($\bar{A}$ and $\bar{C}$) or ($\bar{B}$ and $\bar{C}$)).
4. Now we can use communativity and associativity to drop parenthesiss around things being *or*'ed:
    1. $\bar{A}$ or ($\bar{B}$ and $\bar{A}$) or ($\bar{A}$ and $\bar{C}$) or ($\bar{B}$ and $\bar{C}$).
5. With some extra magic, we now get:
    1. (A and $\bar{B}$ and $\bar{C}$) or ($\bar{A}$ and $\bar{B}$ and $\bar{C}$) or ($\bar{A}$ and B and C) or ($\bar{A}$ and B and $\bar{C}$) or ($\bar{A}$ and $\bar{B}$ and C) or ($\bar{A}$ and $\bar{B}$ and $\bar{C}$) or

$\forall X$ 14-15

Chapter 14: The Very Idea of Natural Deduction

The aim of a *natural deduction system* is to show that particular arguments are valid in a way that allows us to understand the reasoning that the arguments might involve. Unlike truth tables, we manipulate sentences in accordance with rules that we have set down as good rules.

One of the most reasonable reasons why one would use natural deduction is because evaluating truth tables with over 1024 lines (2^10) is unreasonable.

Chapter 15: Basic Rules for TFL

We will develop a *natural deduction* system. For each connective, there will be *introduction* rules that allow us to prove a sentence that has connective as the main logical operator, and *elimination* rules, that allow us to prove something given a sentence that has connective as the main logical operator.

15.1: The idea of a formal proof A *formal proof* is a sequence of sentences. The last line of a formal proof is the conclusion. As an illustration, consider $\neg(A \vee B) \therefore \neg A \wedge \neg B$.

We will start by writing the premise 1. $\neg(A \vee B)$ with a line underneath it. Everything written above the line is an *assumption*, and everything written below the line will either be something that follows form the assumption or a new assumption. Since we hope to conclude that $\neg A \wedge \neg B$, we will conclude our proof with n. $\neg A \wedge \neg B$.

As another illustration, suppose we wanted to consider $A \vee B, \neg(A \wedge C), \neg(B \wedge D) \therefore \neg C \vee D$. The argument has three premises:

1. $A \vee B$
2. $\neg(A \wedge C)$
3. $\neg(B \wedge \neg D)$

   n. $\neg C \vee D$

15.2: Conclusion

If we want to show that Ludwig is both reactionary and libertarian, we can use natural deduction to adopt the following:

- R: Ludwig is reactionary
- L: Ludwig is libertarian

If we were somewhere on lines 8 and 15 for R and L, respectively, then we could write the following proof as follows:

- 8 | R
- 15| L
- $R \wedge L \qquad \wedge I 8, 15$

Furthermore, if you had already proved that Ludwig was both libertarian and reactionary, then you could use *elimination* rules to prove conclude that ludwig is reactionary:

- x | $L \wedge R$
- $R \qquad \wedge Ex$

15.3: Conditional Consider the argument "If Jane is smart then she is fast. Jane is smart $\therefore$ Jane is fast.

Notice how this follows a straightforward conditional elimination rule ($\rightarrow E$):

- m | $A \rightarrow B$
- n | A
- $B \qquad \rightarrow Em, n$

This rule is also sometimes called *modus ponens*. THis is an elimination rule, because it allows us to obtain a sentence that may not contain '$\rightarrow$' having started with a sentence that does contain '$\rightarrow$.'

Now, let's consider the instance where we state: "Ludwig is reactionary. Therefore if Ludwig is libertarian, then Ludwig si both reactionary *and* libertarian."

1	R	
2	$\quad L$	
3	$\quad R \wedge L$	$\wedge$I 1, 2
4	$L \rightarrow (R \wedge L)$	$\rightarrow$I 2–3

Notice how an additional assumption ('Ludwig is libertarian') is necessary for the sake of the argument. To indicate that we're no longer dealing with just our original assumption 'R.' We are now in a position to use $\wedge I$. Now we have shown that, on the additional assumption 'L' that we can obtain '$R \wedge L$' or that we can conclude '$L \rightarrow (R \wedge L)$.' The idea of the rule $\rightarrow I$ is invoked when making additional assumptions.

However, one must be diligent when using these, as we must close the subproof to signify when we have returned to the main proof. Thus, we stipulate that "to cite individual lines when applying a rule, those lines must (1) come before the application of the rule, but (2) not occur within a closed subproof."

1	A	
2	$\quad B$	
3	$\quad\quad C$	
4	$\quad\quad A \wedge B$	$\wedge$I 1, 2
5	$\quad C \rightarrow (A \wedge B)$	$\rightarrow$I 3–4
6	$B \rightarrow (C \rightarrow (A \wedge B))$	$\rightarrow$I 2–5

Biconditional In order to prove $W \leftrightarrow X$ we need to prove X on the assumption W and W on the assumption X. The biconditional rule $\leftrightarrow I$ therefore requires two subproofs. See below.

i	$\quad \mathscr{A}$	
j	$\quad \mathscr{B}$	
k	$\quad \mathscr{B}$	
l	$\quad \mathscr{A}$	
	$\mathscr{A} \leftrightarrow \mathscr{B}$	$\leftrightarrow$I i–j, k–l

Disjunction Suppose that Ludwig is reactionary. Then Ludwig is either reactionary or libertarian. TO say that Ludwig is either reactionary or libertarian is to say something weaker than to say that Ludwig is reactionary.

m	$\mathscr{A}$	
	$\mathscr{A} \vee \mathscr{B}$	$\vee$I m

and

m	$\mathscr{A}$	
	$\mathscr{B} \vee \mathscr{A}$	$\vee$I m

Contradiction and Negation The rule for introducing negation is that we can use it whenever we explicitly contradict ourselves, i.e. whenever we find both a sentence and its negation appearing in our proof:

There are more rules that I couldn't be bothered to include, so read them yourself. Considering what is taught in class almost never overlaps with $\forall X$, don't waste your time unless it's legitimately covered in class.

Logic Rules (Equivalences)

Two expressions are equivalent if they have the same truth value.

Simplifications

They are equivlaences that also work backwards. Here are "the big 5:"

long	simplified	name of rule
$\neg\neg P$	P	double negation
$\neg\top$	$\bot$	definition of $\bot$
$P \wedge \bot$	$\bot$	simplification
$P \wedge \top$	P	simplification
$P \vee \bot$	P	simplification
$P \vee \top$	$\top$	simplification

Here's another important table (remember how to read it!)

operands	$\rightarrow$	$\leftrightarrow$	$\oplus$	$\wedge$	$\vee$
P op P	$\top$	$\top$	$\bot$	P	P
P op $\neg P$	$\neg P$	$\bot$	$\top$	$\bot$	$\top$
$\neg P$ op P	$\neg P$	$\bot$	$\top$	$\bot$	$\top$
$\top$ op P	P	P	$\neg P$	P	$\top$
P op $\top$	P	P	$\neg P$	P	$\top$
$\bot$ op P	$\top$	$\neg P$	P	$\bot$	P
P op $\bot$	$\neg P$	$\neg P$	P	$\bot$	P

Associative and Commutative Properties A binary operators is commutative if its operands can be swapped without changing the meaning of the operation. A binary operator is associative if the pair of them can be re-parenthesized without changing the meaning of their joint operation.

Operator	Associativity	Commutativity
$\neg$	not a binary operator	not a binary operator
$\wedge$	$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$	$P \wedge Q \equiv Q \wedge P$
$\vee$	$(P \vee Q) \vee R \equiv P \vee (Q \vee R)$	$P \vee Q \equiv Q \vee P$
$\oplus$	$(P \oplus Q) \oplus R \equiv P \oplus (Q \oplus R)$	$P \oplus Q \equiv Q \oplus P$
$\rightarrow$	not associative	not commutative
$\leftrightarrow$	$(P \leftrightarrow Q) \leftrightarrow R \equiv P \leftrightarrow (Q \leftrightarrow R)$	$P \leftrightarrow Q$

Note that mixing operators doesn't always hold. Also, it is common to write several operators in a row with/without parenthesis, such as $P \vee Q \vee R \vee S$ instead of $P \vee (Q \vee (R \vee S))$ Parenthesis can be changed with associativity or because they are redundant.

Other equivalences

Form 1	Form 2	Name of Rule
$A \to B$	$\neg A \lor B$	definition of implication
$A \land (B \lor C)$	$(A \land B) \lor (A \land C)$	Distributive Law
$A \lor (B \land C)$	$(A \lor B) \land (A \lor C)$	Distributive Law
$(A \land B) \lor C$	$(A \lor C) \land (B \lor C)$	Distributive Law
$(A \lor B) \land C$	$(A \land C) \lor (B \land C)$	Distributive Law
$\neg(A \land B)$	$(\neg A \lor \neg B)$	DeMorgan's
$\neg(A \lor B)$	$(\neg A \land \neg B)$	DeMorgan's
$(A \leftrightarrow B)$	$A \to B \land B \to A$	definition of implication
$(A \oplus B)$	$(A \lor B) \land \neg(A \land B)$	definition of exclusive or
$A \oplus B$	$\neg(A \leftrightarrow B)$	
$A \leftrightarrow B$	$\neg(A \oplus b)$	xnor
$P \to (A \lor Q)$	$(P \land \neg A) \to Q$	

Entails

Logical Entailment

Given	Entails	Name
$\bot$	x	
	$\top$	
	$A \lor \neg A$	excluded middle
$A \land B$	A	
A and B	$A \land B$	
A	$A \lor B$	
$A \lor B$ and $\neg B$	A	disjunctive syllogism
$A \to B$ and $B \to C$	$A \to C$	hypothetical syllogism; transitivity of implication
$A \to B$ and A	B	modus ponens
$A \to B$ and $\neg B$	$\neg A$	modus tolens
$A \leftrightarrow B$	$A \to B$	
$A \to C, B \to C$, and $A \lor B$	C	
$A \to B, C \to D$, and $A \lor C$	$B \lor D$	
$A \to B$	$A \to (A \land B)$	
$\neg(A \land B), A$	$\neg B$	

Assume-and-Prove entailment A proof assumes that A and derives B entails that $A \to B$. This is commonly used in the inductive step of a proof by induction:

- $A \vdash B$
- $\therefore A \to B$

A proof that assumes A and derives $\bot$ entails that $\neg A$. This is called "proof by contradiction" or an "indirect proof."

- $A \vdash \bot$
- $\therefore \neg A$

A proof $x \in S \vdash P(x)$ entails $\forall_x \in S.P(x)$. This is called "universal induction."

- $x \in S \vdash P(x)$
- $\therefore \forall_x \in S.P(x)$

If P(x) and x is some specific member of S, that entails $\exists_x \in S.P(x)$. This is called "existential instantiation."

- $x \in S$
- P(x)
- $\therefore \exists_x \in S.P(x)$

Set Entailment

Given	Entails
P(x) and $x \in S$	$\exists_x \in S.P(x)$
$\forall_x \in S.P(x)$ and $T \subseteq S$	$\forall_x \in T.P(x)$
$\exists_x \in S.P(x)$ and $T \supseteq S$	$\exists_x \in T.P(x)$

Given	Entails
$\forall_x \in S.P(x)$ and $S \neq \emptyset$	$\exists_x \in S.P(x)$
$\lvert S\rvert \neq \lvert T\rvert$	$S \neq T$
$\lvert S\rvert < \lvert T\rvert$	$S \not\supseteq T$
$\exists_x \in S.P(x)$	$S \neq \emptyset$

Qualified Entailments

Given	Entails	Names
$\exists_x \in S.P(x)$	P(s), for any $s \in S$ we care to pick	universal instantiation
$\exists_x \in S.P(x)$	$s \in S \wedge P(s)$ where s is an otherwise-undefined new variable	existential instantiation
$s \in S \vdash P(s)$	$\forall_x \in S.P(x)$	universal generalization
$P(s) \wedge s \in S$	$\exists_x \in S.P(x)$	existential generalization

Mathematical Identities

The following are all true for all real numbers where both sides of the equal sign are defined:

- $\log_a(a^x) = x$
- $a^{\log_a(x)} = x$
- $\log_a(xy) = \log_a(x) + \log_a(y)$
- $\log_a(\frac{x}{y}) = \log_a(x) - \log_a(y)$
- $\log_a(x^y) = y \log_a(x)$
- $\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$
- $\log_{a^b}(x) = b^{-1} \log_a(x)$

Also the following are true:

- $(a \in \mathbb{Z}) \wedge (a > 1) \vDash (a$ has at least two factors$)$
- $(a \in \mathbb{Z}) \wedge (a > 1) \wedge (a$ has exactly two factors$) \equiv (a$ is prime$)$
- Each integer greater than 1 has exactly one prime factorization

Example Proof: DeMorgan's Law

Theorem 1: For any expressions P and Q, the expression $\neg(P \wedge Q)$ is equivalent to the expression $(\neg P) \vee (\neg Q)$

Option 1: Brute-Force a Truth Table

P	Q	$\neg$	$(P \wedge Q)$	$(\neg P)$	$\vee$	$(\neg Q)$
false	false	**true**	false	true	**true**	true
false	true	**true**	false	true	**true**	false
true	false	**true**	false	false	**true**	true
true	true	**false**	true	false	**false**	false

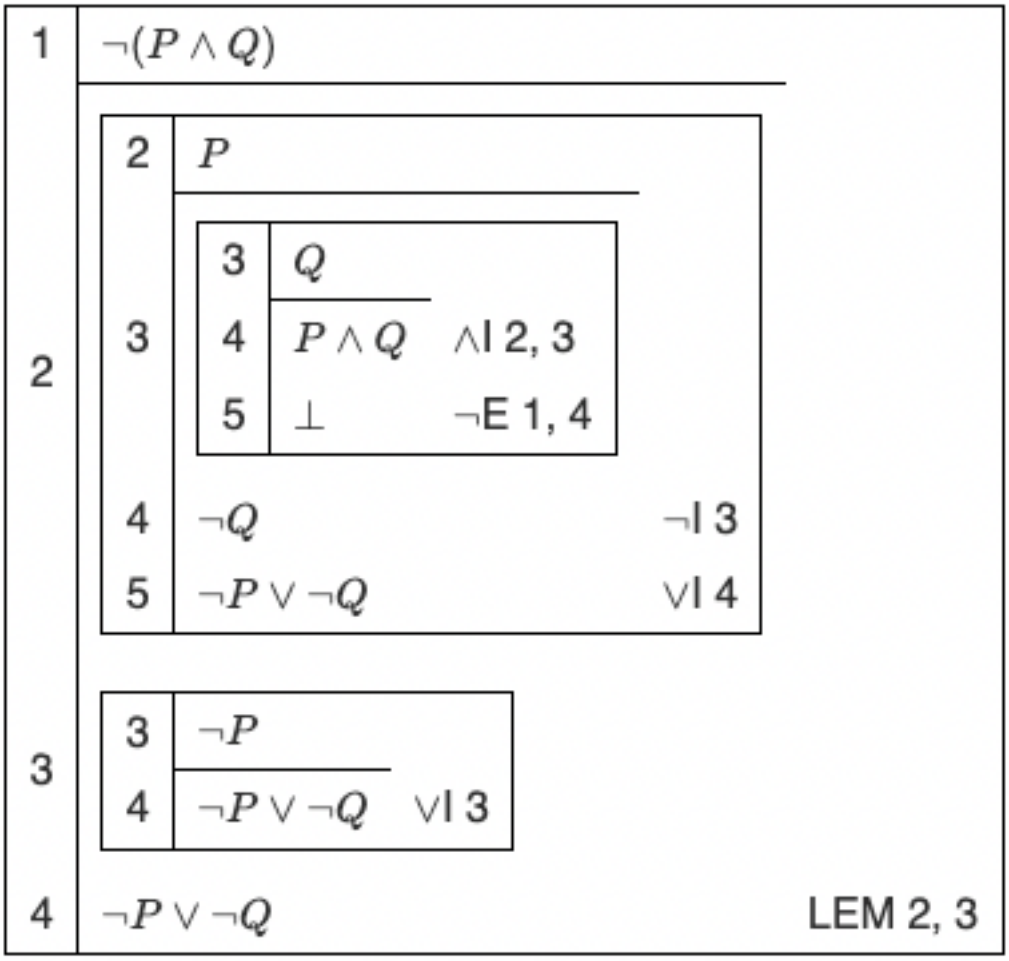
Option 2: Proof by Cases: prose

Proof: Let **N** represent the expression $\neg(P \wedge Q)$ and **O** represent the expression $(\neg P) \vee (\neg Q)$. The proof is by case analysis. There are two cases, either P is true or it is false.

- Case 1: P is true:
  - **N** is $\neg(\top \wedge Q)$; using the identity of *and*, this can be re-written as $\neg Q$. **O** is $(\neg\top) \vee (\neg Q)$, which is equivalent to $\bot \vee (\neg Q)$; using the identity of **or**, this can be re-written as $\neg Q$. Because **N** and **O** are equivalent to the same thing, they are equivalent to each other. Thus the theorem holds in this case.
- Case 2: P is false:
  - **N** is $\neg(\bot \wedge Q)$; using the zero of *and* this can be re-written as $\neg\bot$ which is $\top$. **O** is $(\neg\bot) \vee (\neg Q)$ which is equivalent to $\top \vee (\neg Q)$; using the zero of *or*, this can be re-written as $\top$. Thus, because **N** and **O** are equivalent to the same thing, they are equivalent to each other. Thus, the theorem holds in this case.

Because the theorem holds in all cases, it is true. ■.

Option 3: TFL



TFL Proofs by cases helps demonstrate two important principles:

1. The goal of proof by cases is to add extra information in each case, visible here as the extra given in each sub-proof
2. The cases need to be exhaustive, so that the *or* of all of them is equivalent to $\top$. In TFL, this is ensured by adding specific proof rules (such as LEM) that only apply when this is the case.

Proof Techniques

Apply Equivalence Rules

In a small-step proof, write an equivalent expression and cite the rule used to reach it.

Example — The following uses a note per line to show how it is equivalent to the preceding line

1	$A \vee (B \vee C)$	
2	$(A \vee B) \vee C$	Associative property of $\vee$
3	$(B \vee A) \vee C$	Commutative property of $\vee$
4	$B \vee (A \vee C)$	Associative property of $\vee$
5	$(\neg(\neg B)) \vee (A \vee C)$	Double negation
6	$(\neg B) \rightarrow (A \vee C)$	Disjunction to implication

You can also use prose instead:

> *Example* — This is the same example as the previous one, but written in prose style instead.
>
> $A \vee (B \vee C)$ can be re-written as $(\neg\neg B) \vee (A \vee C)$, which is equivalent to $(\neg B) \rightarrow A \vee C$ by the equivalence of implication and disjuction.

Also apply rearranging - utilizing the associative, communities, and distributive properties of operators. Use simplifying - removing double negation and the ones and zeroes effects of tautologies and contradictions.

Apply Entailment

Because $A \equiv B$ implies $A \vDash B$, you can use equivalence rules in a proof that applies entailment. There are many more entailments (called "proof rules") than equivalence rules, so using them can make a proof construction much easier. A proof using "proof rules" is called a *direct proof*.

(Guide to a) Direct Proof

Axioms

An **axiom** is something we make true without a proof. Sometimes the larger truths used as axioms during a proof are called **logic rules** or **proof rules**.

Direct proof rules are of two types:

1. Equivalences: if $A \equiv B$ then A is true whenever B is true and false whenever it is false.
   - Any expression that is equivalent to true is a **tautology**. Anything that is equivalent to false is called a **contradiction**.
2. Entailments: if $AvDashB$ then A being true is sufficient to establish the truth of B, but not vice versa. B could be true even when A is false.
   - $A \vDash B$ and $A \rightarrow B \equiv \top$ are two ways of saying the same thing
   - if $A \equiv B$ then both $A \vDash B$ and $B \vDash A$.

Strategy

Proofs are like a puzzle - look where you are and pick the piece that makes the most progress. Some tips include:

- See where you are: you want to proceed from the last step, but in some cases you may want to have several chains you combine on later.
- See what can fit: You can always add double negation and always can add on an $\vee\bot$ or an $\wedge\top$, though those are rarely helpful.
- Pick one that makes progress: You want to use the rules that get you closer to the goal. Don't undo what you've done. Turn everything into ands, ors, and work on those. Use DeMorgan. If you don't see anything else that works, pick a step that changes the formula a lot (like distribution) or one that simplifies it.
- Work backwards: When you want to prove that $A \equiv B$, you can start with A and show it is equivalent to B, or you can work backwards; start with B and show it's equivalent to A.

Propositions and *operators*: MCS 1.1 and $\forall x$ 4.3 and MCS3-3.2.0 and $\forall x$ 5 and about $\rightarrow$ and operators

MCS 1.1

Propositions

A *proposition* is a statement that is either true or false. In the examples below, the first is true and the second is false.

- **Proposition 1.1.1** $2 + 3 = 5$
- **Proposition 1.1.2** $1 + 1 = 3$

Being true or false excludes statements that are not binary answers, and also excludes questions whose truth varies on circumstance. Unfortunately, not all propositions are easy:

- **Proposition 1.1.3**: For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.

To check this, we can let $p(n) ::= n^2 + n + 41$ (::= means "equal by definition). However, we see that p(40) $= 40^2 + 40 + 41 = 41 * 41$ which is not prime. But, the point of this was to show that in general, you can't check a claim about an infinite set by checking a finite set of its elements, no matter how large the finite set. And, **Proposition 1.1.3** could also be rewritten as: $\forall_n \in \mathbb{N}$. p(n) is prime.

Some other cool theorems:

- Fermat's Last Theorem: _There are no positive integers x, y, and z such that $x^n + y^n = z^n$ for some integer $n > 2$. This was eventually proved incorrect.

- Goldbach's Conjecture: *Every even integer greater than 2 is the sum of two primes.* To this day, no one nows whether it's true or false.

$\forall_x$ 4.3 - Atomic Sentences

Entirely unhelpful?

MCS 3 - 3.2.0

While there is a ton of ambiguity in the english language in common speech, in mathematics it is necessary to formulate speech precisely, as we can't make an exact argument if we're not sure what the statements we're even arguing mean!

Propositions from Propositions

Propositional variables are used in place of specific propositions and can only take on the values of $\top$ or $\bot$.

3.1.1 - Not, and, and or If P is a proposition, then so is not(P).

P	not(P)
$\top$	$\bot$
$\bot$	$\top$

The truth table for the proposition "P and Q" can be seen below:

P	Q	P and Q
T	T	T
T	F	F
F	T	F
F	F	F

The truth table for the proposition "P or Q" can be seen below:

P	Q	P or Q
T	T	T
T	F	T
F	T	T
F	F	F

If you want to exclude the possibility of having both, then use the "xor," exclusive-or, opeation:

P	Q	P xor Q
T	T	F
T	F	T
F	T	T
F	F	F

3.1.2 - Implies The combinding operation with the least intuitive technical meaning is "implies." This can also be understood as "An implication is true when the if-part is false or the then-part is true." Here is the truth table:

P	Q	P implies Q
T	T	T (tt)
T	F	F (tf)
F	T	T (ft)
F	F	F (ff)

The idea of "implies" can allow us to see whether the proposition evaluates to true or not. For example, "If Goldbach's Conjecture is true, then $x^2 \geq 0$ for every real number x." Now, this sentence is a proposition in the form "P implies Q." The hypothesis, P, is

"Goldbach's Conjecture is true" and the conclusion, Q, is "$x^2 \geq 0$ for every real number x." Since the conclusion is definitely true, we're either on (tt) or (ft) of the truth table - either way, the proposition as a whole is *true*!

Here's an example of a false implication: "If the moon shines white, then the moon is made of white cheddar." Yes, the moon shines white. But no, the moon is not made of white cheddar cheese. So we're on line (tf) of the truth table, and thus the proposition is *false*.

False Hypotheses An illustrative example of false hypotheses is a system specification which consisted of a series of a dozen rules:

- if C_i: the system sensors are in condition i, then A_i: the system takes action i. Or, more concisely, C_i implies A_i for $1 \leq i \leq 12$. Then the fact that the system obeys the specification would be expressed by saying that the :
- [C_1 implies A_1] and [C_2 implies A_2] and ... and [C_{12} implies A_{12}] (3.1)

Suppose conditions C_2 and C_5 are true, and the system indeed takes on the specified actions A_2 and A_5. This means that in this case the system is behaving according to the specification, and accordingly we want the formula 3.1 to come out to *true*. Now the implications C_2 implies A_2 and C_5 implies A_5 are both true because their hypotheses and their conclusions are true. But in order for 3.1 to be true, we need all other implications with the false hypotheses C_i for $i \neq 2, 5$ to be true. This is exactly what the rule for implications with false hypotheses accomplishes.

If and Only If Mathematics joins propositions in one additional way: The proposition "P if and only if Q" asserts that P and Q have the same truth value - either both are true or both are false.

P	Q	P iff Q
T	T	T
F	T	F
T	F	T
F	F	T

for example, the following *iff* statement is true for every real number x: $x^2 - 4 \geq 0 \iff |x| \geq 2$.

- Note that: "iff" is $\iff$.
- For some values of x, *both* inequalities are true. For other values of x, *neither* inequality is true. In every case, however, the IFF proposition as a whole is true.

3.2 - Propositional Logic in Computer Programs

Consider the code snippet: `if (x > 0) || (x <= 0 && y > 100)`. Let A be the proposition taht `x>0` and let B be the proposition that `y > 100`. Then we can rewrite the condition as: "A or (not(A) and B).

3.2.1 - Truth table calculation Let's evaluate the truth table of the expression "A or (not(A) and B)":

A	B	A or (not(A) and B)	A or B
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

Note that somehow, the outputs of A or B are the same as "A or (not(A) and B). Expressions whose truth tables always match are called *equivalent*. So, we can simplify the code without changing its behavior to `if (x > 0 || y > 100)`.

Notation

English	Symbolic Notation
Not(P)	$\neg P$ (alternatively, $\bar{P}$)
P and Q	$P \wedge Q$
P or Q	$P \vee Q$
P implies Q	$P \rightarrow Q$
if P then Q	$P \rightarrow Q$
P iff Q	$P \leftrightarrow Q$
P xor Q	$P \oplus Q$

For example, the notation "If P and not(Q), then R" would be written: $(P \wedge \bar{Q}) \rightarrow R$.

$\forall_X$ 5 - Connections

symbol	what it is called	rough meaning
$\neg$	negation	"it is not the case that"
$\wedge$	conjunction	"both... and..."
$\vee$	disjunction	"either... or..."
$\rightarrow$	conditional	"if... then..."
$\leftrightarrow$	biconditional	"... if and only if ..."

5.1 - Negation Consider the statement B: "Mary is in Barcelona." Using $\neg$, we can symbolize the sentence "It is not the case that Mary is in Barcelona" using $\neg B$.

You can also express things like "The wrench is not irreplaceable" by saying $\neg\neg B$. Be careful of tricks like "Jane is unhappy." If we were to say $\neg B$, then we would be saying "It is not the case that Jane is happy," however the phrase "Jane is unhappy" could mean she could be in a state of neither happiness nor unhappiness.

5.2 - Conjunction Consider:

1. Adam is athletic
2. Barbra is athletic
3. Adam is athletic, and Barbara is also athletic.

The third expression can be written as a conjunction of the two - 2. $\wedge$ 3..

You can then use the previous statements and use parenthesis to express expressions like "it's not the cae that you will get both soup and salad" like $\neg(S_1 \wedge S_2)$ where S_1 means "you will get soup" and S_2 means "you will get salad."

5.3 - Disjunction Consider:

1. Either Fatima will play videogames, or she will watch movies.
2. Either Fatima or Omar will play videogames.
    - F: Fatima will play videogames
    - O: Omar will play videogames
    - M: Fatima will watch movies

To express these expressions, we need to express sentence 1. using $F \vee M$, called *disjunction*. We can express sentence 2 as $F \vee O$.

5.4 - Conditional Consider these sentences:

1. If Jean is in paris, then Jean is in France
2. Jean is in France only if Jean is in Paris.
    - P: Jean is in Paris
    - F: Jean is in France

We can symbolize sentence 1. as $P \rightarrow F$. This connective is called "the conditional." Here, P is the *antecedent* and F is the *consequent*. We can symbolize sentence one as $F \rightarrow P$.

5.5 - Biconditional Consider:

1. Josie is a dog only if she is a mammal
2. Josie is a dog if she is a mammal
3. Josie is a dog if and only if she is a mammal
    - J: Josie is a dog
    - M: Josie is a mammal

Sentence 1 can be symbolized as $J \rightarrow M$. Sentence 2 can be symbolized as $M \rightarrow J$. Sentence 3 says something stronger than either 1 or 2 - it can be paraphrased as "Josie is a dog if josie is a mammal, and josie is a dog only if josie is a mammal. This can be expressed by $J \leftrightarrow M$. This is called the"biconditional," or "iff" - if and only if. A sentence can be symbolized as $A \leftrightarrow B$ if it can be paraphrased in english as 'A iff B,' as 'A if and only if B.'

5.6 - Unless We can use all the connectives to express many kinds of sentences with complex structures. For example, the sentence "Unless you wear a jacket, you will catch a cold" where J="You will wear a jacket" and D="You will catch a cold" can be expressed as $\neg J \rightarrow D$.

About $\rightarrow$: "If" and Logic

If

If it rains, I'll get wet. Logical part is $R \rightarrow W$.

Raining?	Wet?	Consistent with the phrase?
No	no	yes
No	yes	yes
Yes	no	no
Yes	yes	yes

Beyond Logic - there's a connotation which propositional logic cannot encode that rain starts shortly before wetness and there's a causal connotation that propositional logic cannot encode that rain creates wetness.

If you earn a 93% or more, you get an A Logical part: $E \leftrightarrow A$

Earned 93?	Get A?	Consistent with phrase?
No	no	yes
no	yes	no
yes	no	no
yes	yes	yes

Beyond logic - there's a temporal condition - you get the 93% before you get the A - which cannot be encoded in propositional logic. Further, the $\leftarrow$ part is "fuzzy." A 92.8% might get you an A, but an 81% won't. The $\rightarrow$ part is implicitly "or more": an A+ would not be a problem.

This program crashes if you type Ctrl+Q Logical part: $C \leftarrow Q$

it crashes?	Ctrl+Q?	Consistent with phrase?
No	no	yes
no	yes	no
yes	no	yes
yes	yes	yes

Beyond logic - there's a temporal connection that you press Ctrl+Q before it crashes and theres a causal connotation that typing Ctrl+Q triggers the crash that logic cannot encode

When

I love it when it rains Logical part: $L \leftarrow R$

I love it?	It rains?	Consistent with the phrase?
no	no	yes
no	yes	no
yes	no	yes
yes	yes	yes

Beyond logic - "when" connotes that the truth of R comes and goes. It also connotes that rain *will* happen eventually; "if" in this phrase would have some logical meaning, but it would imply doubt that rain would ever occur.

I sing when in the shower Logical part - $\top$

I sing?	Showering?	consistent with phrase?
no	no	yes
no	yes	yes
yes	no	yes

I sing?	Showering?	consistent with phrase?
yes	yes	yes

Beyond logic - This means that there exist some times when i am both in the shower and singing, but does not rule out either singing outside the shower nor that I might have some in-shower time when I'm not singing. We can encode that "there exists some time" idea with first order logic - $\exists t.S(t) \wedge H(t)$ but not with propositional logic.

Only

It only rains at night Logical part - $R \to N$

it rains?	It's night?	consistent with phrase?
no	no	yes
no	yes	yes
yes	no	no
yes	yes	yes

I only love my spouse Logical part - $L \leftrightarrow S$

I love it?	It's my spouse?	Consistent with phrase?
No	no	yes
no	yes	no
yes	no	no
yes	yes	yes

Beyond logic - the use of "only" implies "love" means something more specific than it would like a phrase "I love cake"and possibly something even more specific than "I love my spouse."

Operators

Propositions

Concept	Java/C	Python	This class	Bitwise	Name	other
true	`true`	`True`	$\top$ or 1	-1	tautology	T
false	`false`	`false`	$\bot$ or 0	0	contradiction	F
not P	`!p`	`not p`	$\neg P$ or $\bar{P}$	~p	negation	
P and Q	`p && q`	`p and q`	$P \wedge Q$	p & q	conjunction	PQ, $P \cdot Q$
P or Q	p || q	p or q	$P \vee Q$	P | Q	disjunction	P + Q
P xor Q	`p!=q`	`p!=q`	$P \oplus Q$	p ^ q	parity	$P \veebar Q$
P implies Q			$P \to Q$		implication	$P \supset Q$, $P \implies Q$
P iff Q	`p==q`	`p==q`	$P \leftrightarrow Q$		bi-implication	$P \iff Q$, P xnor Q

Proofs

Concept	Symbol	Meaning
equivalent	$\equiv$	"$A \equiv B$" means "$A \leftrightarrow B$ is a tautology"
entails	$\vDash$	"$A \vDash B$" means "$A \to B$ is a tautology."
provable	$\vdash$	"$A \vdash B$" means "A *proves* B"; it means both "$A \vdash B$ and"I know B is true because A is true. "$\vdash B$" means "I know B is true."
therefore	$\therefore$	"$\therefore A$" means "the lines above this $\vdash A$" and also connotes "A is the thing we wanted to show."
proof done	$\blacksquare_{q.e.d}$	marks the end of a written (prose) proof.
hypothesis		something we expect is true

Concept	Symbol	Meaning
theorem		something we've proven is true
corollary		small theorem that builds off of the main theorem
lemma		small theorem that helps set up the proof of the main theorem

Arithmetic

Concept	Symbol	Meaning
floor	$\lfloor x \rfloor$	the largest integer not larger than x; x rounded down to an integer
ceiling	$\lceil x \rceil$	the smallest integer not smaller than x; x rounded up to an integer
exponent	x^y	x multiplied by itself y times
sum	$\sum_{x \in \mathbb{S}} f(x)$	the sum of all members of $\{f(x) \vert x \in \mathbb{S}\}$. By definition, 0 if S = {}.
Product	$\prod_{x \in \mathbb{S}} f(x)$	The product of all members of $\{f(x) \vert x \in \mathbb{S}\}$. By definition, 1 if S = {}
product	$\prod_{x=a}^{b} f(x)$	$\prod_{x \in \mathbb{S}} f(x)$ where $\mathbb{S} = \{x \vert (x \in \mathbb{Z}) \wedge (a \leq x \leq b)\}$. The product of f(x) applied to integers between a and b inclusive
factorial	x!	$\prod_{i=1}^{x} i$. The product of all positive integers less than or equal to x. The number of permutations of a length-x sequence with distinct members.
choose	$\binom{n}{k}$	$\frac{n!}{(n-k)!k!}$. The number of k-member subsets of an n-element set.

Sets: Sets are Values, Sets vs. Sequences, Sets Writeup

Sets Writeup

Defining Sets

1. A set contains numbers. **set** is thus a term referring to a collection.
   - Member is the word for something inside a set.
2. We write a set with curly braces and commas.
   - {1, 3} is a set. so is {dog, cat, mouse}. Arguably {x?y:z, satisfaction, 2102, :dragon:} is a set but sets whose members do not have some single unifying defined type are so uncommon that many say they're not sets.
3. A member of a set has no other set-related properties besides membership in the set: not order or position in the set, not number of copies in the set.
   - Thus, {1, 2} and {2, 1} are two ways of writing the same set and {1, 2, 2} is nonsensical.
4. The only property of a set is its members
5. A set can be empty
6. A set can be infinite. So is the set of positive integers. . . .
7. A set can have sets as members.
   - {1} is a set, so is {{1}, {}, {84, 5}}, so is the set of all two-element sets of integers.
   - Having sets within sets does not change any other rules - i.e. the set {{1, 2}, {2,1}} is nonsensical.

Set operations and notation

"m is a member of S" is written as $m \in S$.

- Note that $m \in S$ is a predicate: $1 \in \{1, 2, 3\}$ is $\top$ and $1 \in \{2, 4, 6\}$ is $\bot$.

"m is not a member of S" is written as $m \notin S$, that is, $\neg(m \in S)$

- You may also see $\ni$ and $\notin$ which means the same thing as $\in$ and $\notin$ but with the set on the left and the member on the right instead of the other way around.

Sub- and super-sets

$A \subseteq B$ is defined to mean "every member of A is also a member of B," that is, $\forall x.((x \in A) \rightarrow (x \in B))$

- The $\subseteq$ symbol is pronounced "is a subset of." It intentionally looks somewhat like the $\leq$ symbol because if $A \subseteq B$ then A is "less than or equal to B in the sense that everything A has, B has too, and B may have more.

$\top$: True	$\bot$: False
$\{\} \subseteq \{1,2\}$	$\{1,2\} \subseteq \{2,3\}$
$\{1\} \subseteq \{1,2\}$	$\{1,2\} \subseteq \{1\}$

$\{1,2\} \subseteq \{1,2\}$ $\{1,2\} \subseteq$ the integers | $\{1,2.3\} \subseteq$ the integers

$A \subset B$ is defined to mean both "$A \subseteq B$" and "$A \neq B$"; that is, $(A \subseteq B) \wedge (A \neq B)$. The symbol $\subseteq$ is pronounced "is a proper subset of."

$\top$: True	$\bot$: False
$\{\} \subset \{1,2\}$	$\{1\} \subset \{2,3\}$
$\{1\} \subset \{1,2\}$	$\{1,2\} \subset \{1\}$
	$\{1,2\} \subseteq \{1,2\}$
$\{1,2\} \subset$ the integers	$\{1,2.3\} \subset$ the integers

$A \supseteq B$ means $B \subseteq A$; $A \supset B$ means $B \subset A$

- the symbol "$\supseteq$" is pronounced "is a superset of". The "$\supset$" is pronounced "is a proper superset of."

Sets from other sets

$A \cup B$ is defined to mean "A set containing every member of A every member of B and no other members. That is,"a set S such that $\forall x.(x \in S \leftrightarrow ((x \in A) \vee (x \in B)))$"

- The $\cup$ symbol is pronounced "union" and $A \cup B$ is called the union of A and B." Note that $(A \subseteq) \leftrightarrow ((A \cup B) = B)$

A	B	$A \cup B$
{1, 2}	{1, 2}	{1, 2}
{1, 2}	{0}	{1, 2}
{1, 2}	{5, 6}	{1, 2, 5, 6}
{1, 2, 4, 8}	{2, 4, 6, 8}	{1, 2, 4, 6, 8}
the positive numbers	{0}	the non-negative numbers
the even numbers	the odd numbers	the integers

$A \cap B$ is defined to mean "a set containing every member shared both by A and B, and no other members"; that is; "a set S such that $\forall x.(x \in S \leftrightarrow ((x \in A) \wedge (x \in B)))$

- the $\cap$ symbol is pronounced "intersection" and $A \cap B$ is called the "intersection of A and B". It intentionally looks similar to $\wedge$ to suggest a similar value to the member of intersection of A and B if it is a member of A and a member of B.
- Note that $(A \subseteq B) \leftrightarrow ((A \cap B) = A)$

A	B	$A \cap B$
{1, 2}	{1, 2}	{1, 2}
{1, 2}	{}	{}
{1, 2}	{5, 6}	{}
{1, 2, 4, 8}	{2, 4, 6, 8}	{2, 4, 8}
the positive numbers	the integers	the positive integers

If we say that $A \cap B = \{\}$, then we say that A and B are disjoint. Less commonly, **overlap** is used to mean "are not disjoint."

A B is defined to mean "a set containing every member of A that is not a member of B, and no other members"; that is, "a set S such that $\forall x.(x \in S \leftrightarrow ((x \in A) \vee (x \notin B)))$"

- the symbol is pronounced either "Minus" or "set minus." A B has everything A has provided B does not have it. It is rarely used in computing. Note that $(A \subseteq B) \leftrightarrow ((A \backslash B)) = \emptyset$

A	B	$A \backslash B$
{1, 2}	{1, 2}	{}
{1, 2}	{}	{1, 2}
{1, 2}	{5, 6}	{1, 2}
{1, 2, 4, 8}	{2, 4, 6, 8}	{1}
the integers	the positive numbers	the negative integers

pow(A) or P(A) is defined to mean "the set of all subsets of A"; that is, "a set S such that $\forall x.(x \in S \leftrightarrow (x \subseteq A))$". pow(A) or P(A) is called the power set of A and is the set of all subsets of A. Note that every set, even {}, has one subset: {}; thus, a power set is never empty, $\emptyset \in P(A)$ is a tautology, and $\emptyset = P(A)$ is a contradiction regardless of what set A is.

A	P(A)
{}	{{}}
{1}	{{}, {1}}
{1, 2}	{{}, {1}, {2}, {1, 2}}
{{1, 2}, {3}}	{{}, {{1, 2}, {3}, {{1, 2}, 3}}
{{}}	{{}, {{}}}

Concept	Set Notation	Actuality
"add" x to S	$S \cup \{x\}$	a set like S except it also has x in it
"remove" x to S	$S \backslash \{x\}$	a set like S except it does not have x in it.
A "xor" B	$(A \cup B) \backslash (A \cap B)$	a set with elements in A or B but not both

Counting Members

$|A|$ means "the number of distinct values that are members of A." We call this notion the **cardinality** and read $|A|$ as "the cardinality of A."

A	$\|A\|$
{}	0
{1, 2}	2
{{1, 2}}	2
{1, {2, 3}}	2
P({1, 3, 5, 7})	16
P(A)	$2^{\|A\|}$

Set-builder notation

Set-builder notation $\{x \in \mathbb{Z} | x^3 - 30x + 1 > 0\}$ would be written: "x for x in Z if $x^3 - 30x + 1 > 0$", although you'd have to define a sensible, finite Z range.

Common sets

There are some common sets that have their own symbols. Many of them are listed in MCS 4.1.1

Symbol	Set	Elements
	the empty set	none
$\mathbb{N}$	nonnegative integers	{0, 1, 2, 3, ...}
$\mathbb{Z}$	integers	{..., -3, 2, -1, 0, 1, 2, 3, ...}
$\mathbb{Q}$	rational numbers	$\{\frac{1}{2}, -\frac{5}{3}$, 16, etc.$\}$
$\mathbb{R}$	real numbers	$\{\pi, e, -9, \sqrt{2}, etc\}$
$\mathbb{C}$	complex numbers	$\{i, \frac{19}{2}, \sqrt{2} - 2i, etc\}$

Quantifiers and Sets

Quantifiers are often used with sets. Set-notation quantifiers and domain-bound quantifiers can be each defined in terms of the other.

Core Notation

The notation $\forall x \in S.P(x)$ means "the predicate P(x) is true for every x in the set S." It does not say anything about the true or falsehood of P(x) for x not in S, nor does it assert that there are any members of S.

The notation "$\exists x \in S.P(x)$" means "there is at least one element of S, and at least one element of S makes the predicate P(x) true. It does not say anything about the truth or falsehood of P(x) for x not in S, nor if there are more (or even at all) members of S that also make P(x) true.

- "$\forall x, y \in S$" is shorthand for "$\forall x \in S.\forall y \in S$".
- "$\exists x, y \in S$" is shorthand for "$\exists x, y \in S.\exists y \in S$".

Converting "$\forall x \in S...$" to "$\forall x...$"

If a domain is not specified and all quantifiers are given with sets, the implicit domain is union of all such sets or any superset containing that union.

Converting "$\forall x....$" to "$\forall x \in S...$"

Define a set U representing the entire domain. The symbol U is not required, but is often used with the intent that it suggest the "universal set" or the "universe of discourse."

Sets vs Sequences

Sets ands sequences are "composite" values because they are made up of other values - true, false, and numbers. Both sets and sequences:

- May contain any other value, including other sets and sequences
- Are values, not entities
- are used to define other structures of interest in discrete math
- are commonly written with their contained values separated by commas

Set:

- Written with curly braces, like {1, 2}
- Cannot contain the same value more than once
- Members have no order, i.e. {2, 3} = {3, 2}
- The empty set (the only set with cardinality 0) is written {} or $\emptyset$
- Has many operators and special notations like $\{1, 2\} \cup \{x^2 | x \in \mathbb{N}^+\}$
- A singleton set is always distinct from its member; $\{2\} \neq 2$
- always called a "set"
- contained values are called "members"; "element" is also sometimes used

Sequence

- Written with parenthesis, like (1, 2)
- Can contain the same value any number of times; (1, 1) is a sequence and is distinct from (1) and (1, 1, 1)
- Items have an order, i.e. (2, 3) $\neq$ (3, 2)
- the empty sequence (the only sequence with length 0) is written () or ϵ
- has no operators that are commonly used in computing
- A singleton sequence is often considered equal to its item: i.e. (2) = 2
- Called a "sequence" or a "tuple", with special words for some lengths (i.e. "pair" or "triple") and some element types (i.e. "string")
- contained values are called "items"; "elements" is also sometimes used

Sets are Values

By example 7 is a value. Charlie is an entity. If we let x and y both refer to 7 and make x bigger, x is no longer 7. This gets to the idea that 7 is a value: an unchanging Platonic ideal. It is impossible, by definition, to make 7 itself bigger. Similarly, if Charlie eats a ton and grows, he's still Charlie.

Because x and y both refer to a value, changing the value of x has no impact on y whereas "Charles" and "Charlie" both refer to an entity, so making "Charlie" richer means "Charles" is richer.

Entities and Values

A value can be written in multiple ways "3+4" and "7" represent the same *integer* but "3+4" and "7" represent different *arithmetic expressions*.

Variables name just one value within a given context Within a single problem, if one assigns $x=2$, then x=2 and this cannot change.

Discrete math values have related programming entities Sets are values in discrete mathematics, just like numbers are: unchangeable Platonic ideals. Everything in discrete mathematics is a mathematical construct and thus a value, not an entity.

Good Proofs in Practice

The purpose of a proof is to establish the truth of an assertion with absolute certainty. To be understanding and helpful, a proof should not only worry about correctness, but also be clear. here are some helpful tactics to achieve this:

- State your game plan: a proof begins by explaining the general line of reasoning, for example "we use case analysis" or "we argue by contradiction."
- Keep a linear flow: proofs are written like mathematical mosaics.. the steps of an argument should follow one another in an intelligible order.
- A proof is an essay: a proof looks like an essay with some equations thrown in.
- Avoid symbolism: use words when you can!
- Introduce notation thoughtfully: An argument can be easily simplified with a variable or defining a new term, but do this sparingly because the reader then has to remember all of that!
- Structure long proofs: Use preliminary lemmas to cite repeatedly.
- Be wary of the obvious: what is obvious to one reader might not be to another - you don't have to prove every single claim you say, but don't use phrases like "clearly" or "obviously" to bully your reader into thinking they're dumb or accept it as true just because

Skills from Mathematics

1. Discussing definitions
    1. Mathematicians begin with thinking hard about definitions early in their undergraduate career and fluency in discussing definitions is something that can benefit everyone.
2. Coming up with counterexamples
    1. When coming up with a new definition, one has as et of examples and counterexamples that one wants the definition to adhere to. So examples and counterexamples help build good definitions.
    2. When encountering an existing definition, the first thing a mathematician does is write down examples/counterexamples.
3. Being wrong and often admitting it
    1. Fostering doubt, being wrong, admitting it, and starting over distinguishes mathematical discourse even from much praised scientific discourse. There's no fame or money behind it, just the personal insight for truth.
    2. The mathematical habit is putting your personal pride or embarrassment aside for the sake of insight.
4. Evaluating many possible consequences of a claim
    1. The limits of an argument result in an even better and more elegant theorem that includes the original claim. More often, you realize you were wrong. So this habit is a less formal variation of being wrong often and coming up with counterexamples.
5. Teasing apart the assumptions underlying an argument
    1. Treating claims mathematically rather than through emotion allows us to tease apart claims without bias.
6. Scaling the ladder of abstraction
    1. Mathematicians "scale the ladder" of abstraction - i.e. they start at the lowest rung where they understand some examples of a paper, jump to the main theorem of the paper, and then synthesize that information to the real world.