Math 270A Study Guide

NOTE - This is intended for people who understand C++ syntax and boolean logic. If you are confused, use this chart to help deduce your problem. Good luck.

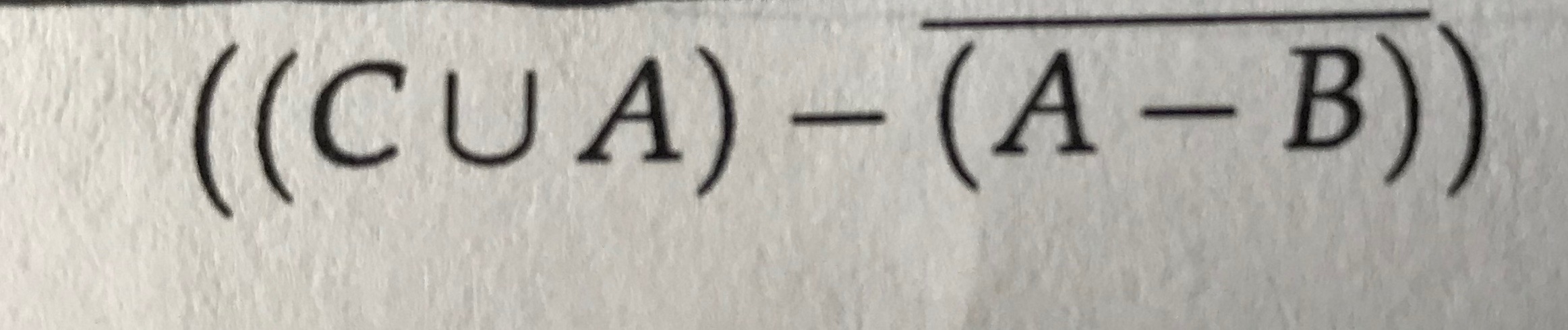
|  |  |  |
| --- | --- | --- |
| Symbol | Meaning | English |
| && | ^ | Conjunction |
| || | (upside down) ^ | Disjunction |
| {} | This is a brace expansion notation introduced in bash shell scripting:  Example : set A = {1-9}  Set A now contains all values from 1 through 9.  Example 2:  mkdir -p Hello/{World,Mom,Dad,Dog}  Ls :  Hello/World  Hello/Mom  ….  Sub folders are made |  |

Quizzes

# Quiz 2

1. Determine the truth value for the following statements:
   1. ∀x∈ℝ, ∀y∈ℝ(x2 ≤ y+1)
      1. False. Let x = 10 and let y =1. **100 ≤ 2 is incorrect.**
   2. ∀x∈ℝ, ∃y∈ℝ(x2+y2 = 9)
      1. Placeholder
2. Negate the following proposition
   1. ∀x∈ℤ(x≥2)
      1. Placeholder
3. Use DeMorgan’s Law to negate the following conjunction:
   1. All accountants own porches and all math teachers own Volkswagens.
      1. Placeholder

# Quiz 3

1. Given N = {A, B, C}. Write out the complete set of P(N)
   1. P(N) = { {A}, {B}, {C}, {A,B}, {A,C}, {B,A}, {B,C}, {C,A}, {C,B} }
2. If A⊆B and B⊆A, what must be true?
   1. That both sets of A and B are the same because A⊆B means that A is in B. Also, B⊆A states that B is in set A. Therefore A and B are the same.
3. Draw a Venn diagram of : 
   1. Placeholder!

# Quiz 4

1. Prove for all integers, m and n if m and n are odd, then mn must be odd
   1. Let m = 2x + 1
   2. Let n = 2y + 1
   3. (2x+1) \* (2y+1)
   4. 4xy+2x+2y+1
   5. 2(2xy+x+y)+1 ← is odd because whichever value that you get from the parenthesis will be multiplied by 2 and added by one. This is the definition of an odd number
2. If for all integers n, if n2+1 is even, then n is odd:
   1. p → q
   2. ! q → ! p
   3. If n is even then n2+1 is odd
   4. Let n = 2x , (2x)2+1
   5. 4x2+1
   6. 2(2x2)+1
   7. Because the contrapositive is true, ,the original statement is true

# Chapter 2

**Section 2.1**

Proposition - A statement that is either true or false but **cannot** be both

Example - (A == B)

Counterexample - (A == B && A != B)

Negation - Opposition of a proposition

Example - return bool (! operator==(const object a, const object b))

Truth Table : A table which lists all possible outcomes of our proposition

|  |  |
| --- | --- |
| P | !P |
| 1 | 0 |
| 0 | 1 |

Conjunction - AND Statement

|  |  |  |
| --- | --- | --- |
| P | Q | P && Q |
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Example - At dinner you get to have water AND a meal

Disjunction - OR Statement

|  |  |  |
| --- | --- | --- |
| P | Q | P || Q |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Example - You can either get a soup OR a salad at dinner, not both

Set - A collection of objects (data structure)

Number Sets

ℕ - natural numbers (non-negatives)

ℤ - integers

ℝ - real numbers (including zero) (non-imaginary)

ℝ\* - real numbers excluding zero (non-imaginary)

ℚ - rational numbers

**Section 2.2**

Element - a member of a set

s ∊ A → s is an element of A

NOTE - Do **not** use mathematical notations as abbreviations in writing

Example : “x and y” != “x && y”

Exercises

1. X and Y are rational
   1. X,Y ∊ ℚ
2. (5 > 1) && (7 < 1)
   1. False
3. (5 == 1) || (7 == 5)
   1. False
4. ( 5 < 1 ) && (7 < 1)
   1. False
5. ( 5 == 5) && (7 == 7)
   1. True

**Section 2.3**

Implication - a conditional proposition is a pair of propositions, one being the hypothesis and the other being the conclusion. These are generally if-else statements.

Denotation - **P → Q**

Meaning : P implies Q only if Q, if P, then Q.

|  |  |  |
| --- | --- | --- |
| P | Q | P → Q |
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | 0 | 1 |

Example : Assume P and R are true. Q is false.

1. (P && Q) → R = true
2. (P || Q) → ! R = false
3. P && (Q → R) = true
4. P → (Q → R) = true

Counterexample

X^2 = 4, then x = 2. This is false because x can equal -2 OR 2.

Converse : **Q → P** (Q implies P)

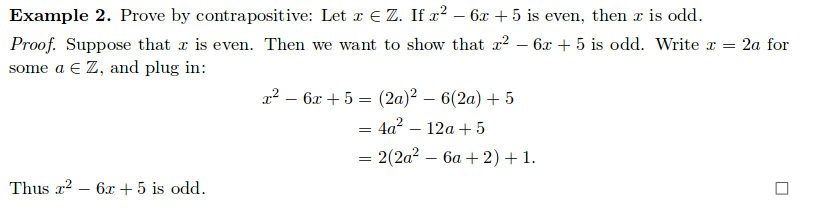
|  |  |  |
| --- | --- | --- |
| P | Q | Q → P |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Inverse : **!P → !Q** (inverse P implies inverse Q)

|  |  |  |
| --- | --- | --- |
| P | Q | !P → !Q |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Contrapositive : **!Q → !P** (inverse Q implies inverse P)

|  |  |  |
| --- | --- | --- |
| P | Q | !Q → !P |
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | 0 | 1 |

Example : 

Logically Equivalent: When two truth tables are identical

Necessary Condition: For some affairs “S”, is a condition that must be satisfied in order to obtain “S”

Sufficient Condition: For some state of affairs “S” is a condition that, if satisfied, guarantees that “S” is obtained.

**Section 2.4**

Biconditional Statements : All inputs must be the same truth value (either **tautology** or **contradiction**)

|  |  |  |
| --- | --- | --- |
| P | Q | P ←→ Q |
| 1 | 1 | 1 (AND GATE) |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 (NAND GATE) |

**Section 2.5 (Continuation)**

Logical Equivalences

* Contradiction : **Always** false
* Tautology : **Always** true
* Symbol : **≡** (three bar equal sign)

|  |  |  |  |
| --- | --- | --- | --- |
| P | !P | P || !P | P && !P |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |

Contingency: Can either be true or false (anything that is not a tautology **or** a contradiction)

Properties:

* Commutative : (P || Q) ≡ (Q || P)
* Associative : (P{&&,||}Q ){&&,||} R ≡ P{&&,||} (Q {&&,||} R)
* Distributive : P || (Q && R) ≡ (P || Q) && (P || R)
* De Morgan’s : !(P || Q) → (!P && !Q)
* Inverse:
* Identity
* Domination

**Section 2.6**

Logical Quantifiers: a way to state that a certain number of elements fulfill some criteria

* Universal: For every possible value/element in a set, we apply the function
  + Symbol : ∀
  + Every single value **must** be true (tautology)
  + To disprove a UQ, we use a counter example as only one is needed to invalidate the claim
* Existential: There exists at least one element in a set which satisfies the function
  + To prove, find one example
  + Hard to disprove : all cases in which it does fail

# Chapter 3

Direct Proof : Hypothesis that is assumed to be true using material previously learned to arrive to a conclusion that is true. Thus P => !Q

Example #1 : Prove the sum of two integers that are odd is even.

* Even : {n ∊ ℤ | 2n is even}
* Odd : { n ∊ ℤ | 2n+1 is odd }
* m,n ∊ ℤ
* 2n + 2m + 1 + 1
* 2(n+m+1)
* Conclusion: Since (n+m+1) is an integer, 2(n+m+1) is even

Example #2 : Let A divide B and B divide C. Then A divide C.

* B = a \* n → n ∊ ℤ
* C = b \* m → m ∊ ℤ
* C = (A \* n)m
* = A(m\*n)
* Conclusion : Since “nm” is an integer, A divides B.

Proof by Contradiction : Assume opposite of the conclusion is true and work backwards to the hypothesis. If we reach a contradiction, we know our assumption is false, so the contradiction is false.

Example #1 :

**Section 3.4 : Mathematical Induction**

Mathematical induction can be used to prove that a statement about *n* is true for *n*≥1

Steps

1. Basis - Verify P(1) is true
2. Inductive - Show that if P(k) is true for some integer k ≥1, then P(k+1) is also true

# Chapter 4

## **Section 4.1**

Sets: Collections of objects which are called **elements**

Example:

ℕ - natural numbers (non-negatives)

ℤ - integers

ℝ - real numbers (including zero) (non-imaginary)

ℝ\* - real numbers excluding zero (non-imaginary)

ℚ - rational numbers

Roster Method: List out all elements in a set

Example: All natural numbers not exceeding 7

* {1, 2, 3, 4, 5, 6, 7}

Set Builder Notation: Shorthand for roster method

Example: All natural numbers not exceeding 7

* { x∈ℤ | x > 7 }
* **|** ← “such that” or “for which”
* { pattern | membership }

## **Section 4.2 : Subsets and Power sets**

Universal Set: The collection of all the possible objects under consideration

Example:

U - Set of Geometric Figures

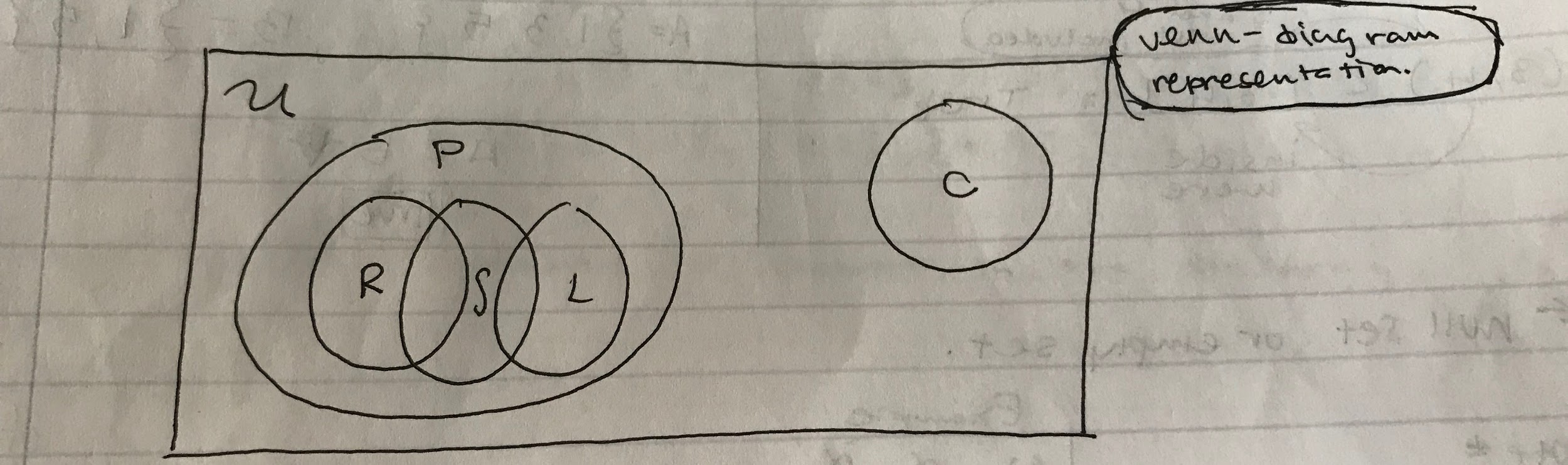
S - Set of Squares

P - Set of Parallelograms

R - Set of Rhombuses

L - Set of Rectangles

C - Set of Circles



Superset: Given a set A and set B, A is a subset of B if every element in A is also in B. Only if every element of A belongs in B but B **does not** have to be in A.

* Denoted as : A ⊆ B



Example:

* A = {1, 2, 3}
* B = {1, 2, 3, 4, 5}
* A ⊆ B because A contains “1, 2, 3” and B also contains “1, 2, 3”

Proper Subset: If A is a subset of B and A != B

* Denoted as : A ⊂ B

Example:

* Let A = (3, 4), let B = [3, 4]
* A ⊂ B
* 3 and 4 are not in A
* 3 and 4 are in B

Null set: Ø

* Note: Ø != {Ø}
* {Ø} == {{}}
* Example:

1. Ø ∈ Ø : False, null cannot be an element of itself
2. 1 ⊆ {1} : False, an element cannot be compared to a set directly

Power Set: Given set A, the power set of A is the set of all possible subsets of A (permutations). This is denoted a P(A). Ø is always a subset of any set. If A is an n-element set, then P(A) has **2n** elements.

Example: Let A = {P, Q, R}

P(A) = {Ø, {P}, {Q}, {R}, {P,Q}, {P,R}, {Q,R}, {P,Q,R}}

Note: A set is said to be finite if it has a finite number of elements

Cardinality: Amount of elements in the set

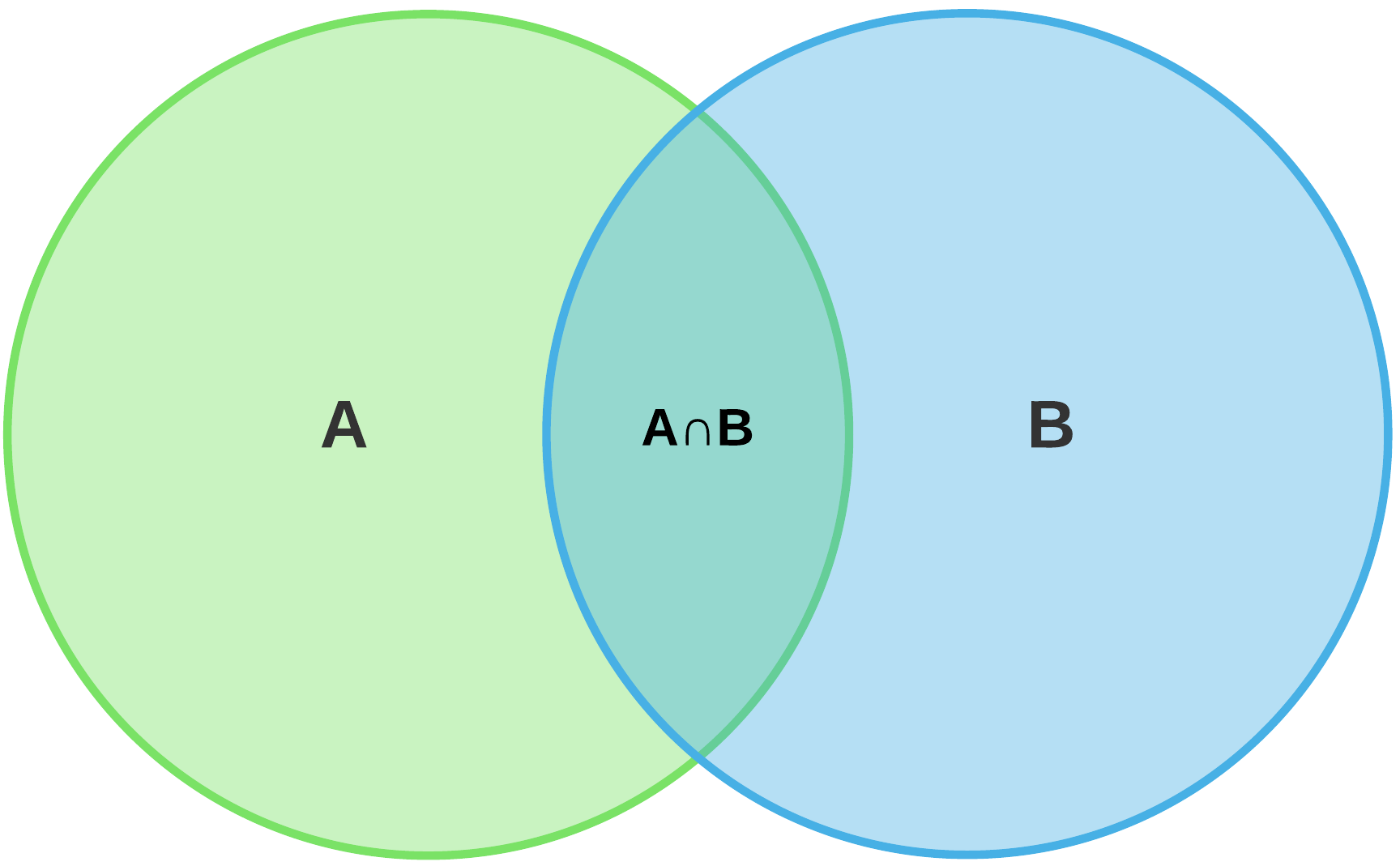
Examples:

| {1, 2, 3, 4} | = 4 | {0, 1} | = 2

| { 0 , {1}} | = 2 | {{1}, 2, 3, 4} | = 4

Intersection: Given two sets A and B

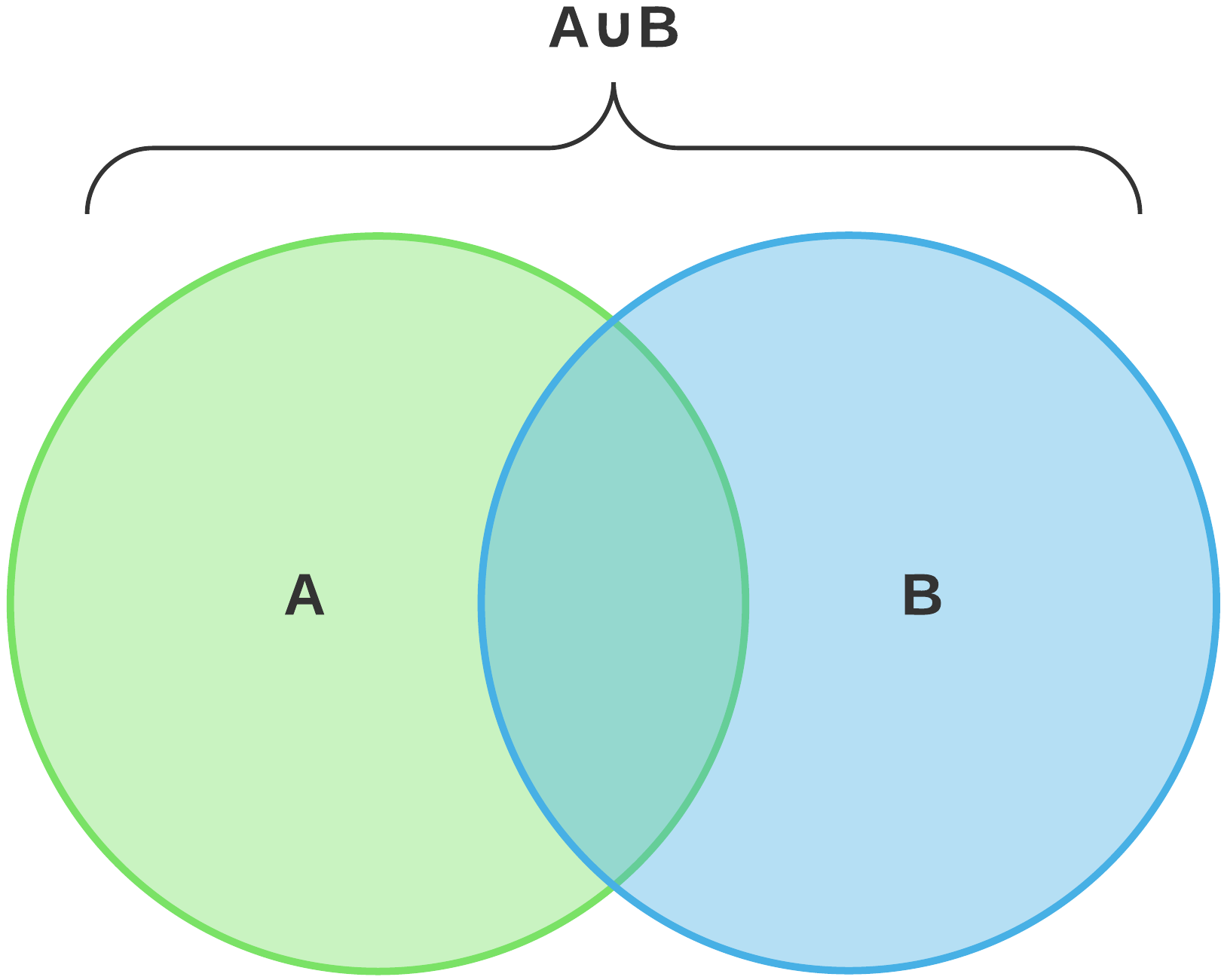




Not A or B, but the elements common between the two.

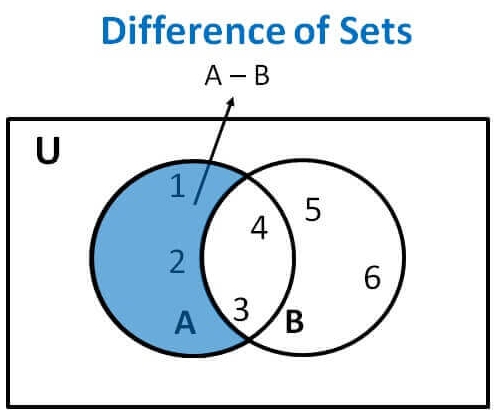
Union: Given two sets A and B





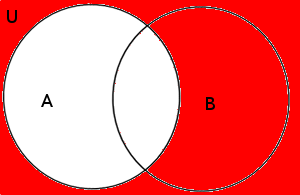
Set Difference:





Found in A but not in B. x is **not** an element of B!!

Ā = U - A



Found in the universal set but nothing from A

**Section 4.4 : Cartesian Products**

* (a, b) != (b, a) unless a == b
* The cartesian product of A & B is the set
  + A x B = {(a, b) | (a ∈A) && (b ∈B)}
    - Contains all the ordered pairs in which the first elements are selected from A & the second elements of B
  + Example:
    - Let A = {1, 2} and let B = {2, 5, 6}
    - A x B = { (1, 2), (1, 5), (1, 6), (2, 2), (2, 5), (2, 6) }
    - A x A = { (1, 1), (1, 2), (2, 1), (2, 2) }
  + If A & B are finite sets, then | A x B | = |A| \* |B|

# Chapter 6

**Section 6.1**

Introduction - We call the result from “F” the image of x under “F”, f(x) also known as “F of x”.

Domain - Set of input values

Codomain - Set of possible output values

* Possible because not every element in the codomain needs to appear as the image of some element from the domain
* The **range** of the function is the collection of image/s that formed from the codomain
* The range is also the proper subset of the codomain
* If the range of a function does equal the codomain, it is onto.

**Section 6.2**

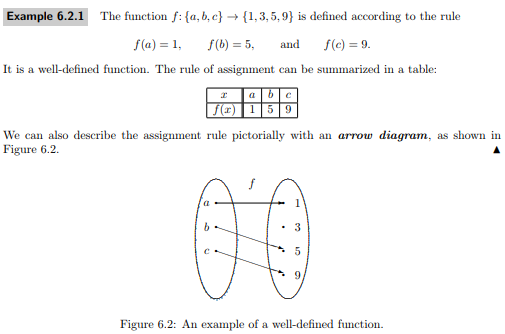
Let *A* and *B* be nonempty sets:

* A **function** from *A* to *B* is a rule that assigns to every element of *A* a unique element in *B*.
* We call *A* the **domain** and *B* the**codomain** of the function.

A function is sometimes called a **map** or **mapping.**

* We sometimes say that *f* **maps** *x* to its image *f(x)*
* If the function is called “f”, write f : A → B; where A is the domain and B is the codomain
* Every element in the domain has an image under f and the image is unique.
* ^ x→ f(x)

Example



The two key requirements of a function are:

* every element in the domain has an image under *f*
* the image is unique

**Section 6.3 : One-to-one Functions**

Two special families of Functions

1. One-to-one
2. Onto

One-to-one

* A function *f* is one-to-one if *f*(x1) = *f*(x2) → x1 = x2
* Also called injective
* x1 != x2 → *f*(x1) != *f*(x2) OR *f*(x1) = *f*(x2) → x1 = x2

Example:

# Things in limbo

## Venn Diagram drawings:

