1.1 Proofs

Double Negative	not(not(P)) <=> P
Associative Law	(X and (Y or Z)) <=> ((X and Y) or Z)
Distributive Law	(X and (Y or Z)) <=> ((X and Y) or (X and Z)
Contrapositive	X → Y <=> not(Y) → not(X)
Equivalence	X ↔ Y <=> (X → Y) and (Y → X)

Proof Techniques
Exhaustive checking
Conditional proof
Proof by contradiction

1.2 Sets

Natural Numbers (N)	[0], 1, 2, 3...
Integers (Z)	…, -3, ,-2, -1, 0, 1, 2, 3...
Rational numbers (R)	A / B, A and B in Z

Two characteristics of sets:
There are no repeated occurrences of elements
There is no particular order of elements

Power sets
S = {a, b, {a, c}}
power(S) = {NULL, {a}, {b}, {{a, c}}, {a, b}, {a, {a, c}}, {b, {a, c}}, {a, b, {a, c}}}
Cardinality(S) = |S| = 3
Subsets:
A is a subset of B if A contains all elements in B
A is a proper subset of B if A is a subset of B and A != B
Two sets are equal if they have the same elements, so A is a subset of B and B is a subset of A
Set Operations
A ∪ B = { x | x ∈ A or x ∈ B }
A ∩ B = { x | x ∈ A and x ∈ B }

Set properties

A ∪ NULL = A
A ∪ (B ∪ C) = (A ∪ B) ∪ C
A – B = {x | x ∈ A and x not ∈ B}
A' = U – A
A ∪ (A ∩ B) = A, A ∩ (A ∪ B)
| A ∪ B| = |A| + |B| - |A ∩ B|
(A ∪ B ∪ C ∪ D)' = A' ∪ B' ∪ C' ∪ D'

1.3 Ordered Structures
Tuple: An ordered collection of elements
Two characteristics of tuples:
There may be repeated occurrences of elements
There is an order or arrangement of the elements

A = {A, B, C}, B = {1, 2}
A^0 = {()}
A^1 = {(a), (b), (c)}
A^2 = {(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)}
A * B = {(a, 1), (a, 2), (b, 1), (b, 2), (c, 1), (c, 2)}

List: a finite ordered sequence of zero or more elements.
Two characteristics of lists:
Elements can be repeated
Only two accessible elements: element at the head, and the tail which is the list of elements following the head
Cartesian Products

cons(h, L) creates a list whose head is h and whose tail is L

1.4 Graphs

Graph: a set of nodes that are interconnected by edges
Adjacent: two nodes share an edge
n-colorable: if a graph is n-colorable then its edges can be assigned n colors without any adjacent nodes sharing two same colored edges
Chromatic number: the mnimum n-color for a graph
Complete graph: every node has an edge to every other node
Path: the series of edges that link one node to another
Connected graph: all nodes have a path to every other node
Cycle: a path that starts and ends at the same node

Graph traversal
Breadth first: visit all unvisited adjacent nodes of a given node, then visit all unvisited nodes adjacent to the adjacent nodes
Depth first: for some node, visit unvisited node, visit rest of unvisited nodes.

Tree: a connected graph without cycles
Rooted tree: a tree with a node designated as root
Height: the number of edges from root to farthest child
Leaf: a childless node

2.1 Functions

Function: an association between two sets, A and B, that map exactly one element from set A to set B.

Given sets A and B and function f: A → B

f maps elements from A to B
Domain: the set A
Codomain: the set B
Range: the subset of B that is mapped to by A
range(f) = {f(a) | a ∈ A }
Image: For any set S that is a subset of A, the image is the elements in A that are actually mapped to B
Injection, one-to-one: if f maps distinct elements of A onto distinct elements of B. AKA Differentiable.
Surjection: if f has a value from A mapped to every value in B. Also, range(f) = codomain(f)
Bijection: f is injective and surjective
EG
A = {a, b, c}, B = {1, 2, 3}
f: A → B

f(a) = 1
f(b) = 1
f(c) = 2

domain(f) = {a, b, c}
codomain(f) = {1, 2, 3}
range(f) = {1, 2}

Images:
f({a}) = {1}
f({a, b}) = {1}
f({a, b, c}) = {1, 2}

Preimages:
f^-1({1, 3}) = {a, b}
f^-1({3}) = NULL

Partial functions: Functions that are undefined for some values

2.1 Various functions

Floor(x) => f: R → Z
Floor(R) = Z iff R <= Z < R + 1 iff Z - 1< R <= Z
Properties:
Floor(R + Z) = Floor(R) + Z
Floor(R) < Z iff R < Z
Z <= Floor(R) iff Z <= R

Ceiling(x) => f: R → Z
Ceiling(R) = Z iff R – 1 < Z <= R iff Z <= R < Z + 1
Properties:
Ceiling(R + Z) = Ceiling(R) + Z
Z < Ceiling(R) iff Z < R
Ceiling(R) <= Z iff R <= Z

gcd(a, b) = gcd(b, a)
gcd(a, b) = gcd(b, a – bq) q ∈ Z
if g = gcd(a, b), then g = ax + by, x, y ∈ Z

algorithm:

a = bq + r, r ∈ Z, b != 0

2.4 Countability

Given sets A and B, if A biject B, then |A| = |B|

Informally, a set is countable if its elements can be counted in a step by step manner.
Formally, a set is countable if it is finite or there is a bijection between it and N.

Countable properties:
Every subset of N is countable
S is countable iff |S| <= |N|
If S0 .. SN is a sequence of countable set, S0 ∪ .. ∪ SN is countable.

Diagonalization:

Let A be an alphabet with two or more symbols and let S0 .. SN be a countable listing of sequences. The sequences are listed as the rows of an infinite matrix

S0| a0, a1, a2, a3, a4, a5, a6,
S1| a10, a11, a12, a13, a14, a15, a16,
S1| a20, a21, a22, a23, a24, a25, a26

Then there is a seqeuence S = (a0, a1, a2, a3, a4, …) over A that is not in the original list. S can be constructed from a diagonal list of elements, (a00, a11, a22, a33, …).

3.1 Inductively defined sets

An inductive definition of a S set consists of three steps:
Basis: Specify one or more elements of S.
Induction: Give one or more rules to construct new elements of S from existing elements of S.
Closure: State that S consists exactly of the elements obtained by the basis and induction steps (assumed).

All Strings over A
Basis: Λ ∈ A*
Induction: if s ∈ A and a ∈ A, then as ∈ A*.

3.2 Recursive functions and procedures
If S is an inductively defined set, then we can construct a function f with domain S as follows:
1) For each basis element x ∈ S, specify a value for f(x).
2) Give rules that, for any inductively defined element x ∈ S, will define f(x) in terms of previously defined values of f.

3.3 Grammars

A grammar is a set of rules used to define the structure of the strings in a language.

If L is a language over an alphabet A, then a grammar for L consists of a set of grammar rules of the form a → b, where a and b denote strings of symbols taken from A and a set of grammar symbols disjoint from A.
The a → b notation is also known as a production.

The four parts of a grammar
1) An alphabet N of grammar symbols called nonterminals
2) An alphabet T of symbols called terminals. Distinct from N.
3) A specific nonterminal S, called the start symbol.
4) A finite set of productions of the form a → b, where a and b are strings over the alphabet N ∪ T with the restriction that a is not the empty string.

EG:
S → Λ | aS | bS | cS
P = { S → Λ, S → aS, S → bS, S → cS}
4 tuple = ({S}, {a, b, c}, S, P}

4.1 Properties of binary relations

For a binary relation R on a set A, we have the following definition
1) R is reflexive if x R x for all x ∈ A
2) R is symmetric of x R y implies y R x for all x, y ∈ A
3) R is transitive if x R y and y R z implies x R z for all x, y, z ∈ A
4) R is irreflexive if (x, x) !∈ R for all x ∈ A
5) R is antisymmetric if x R y and y R x implies x = y for all x, y ∈ A

If R and S are binary relations, then the coposition of R and S is the following relation:
R composition S = {(a, c) | (a, b) ∈ R and (b, c) ∈ S}

4.2 Equivalence relations

Any binary relation that is reflexive, symmetric, and transitive is called an equivalence relation.
Intersection Property of Equivalence
If E and F are equivalence relations on the set A, then E ∩ F is an equivalence relation on A.

Kernel relations
If f is a function with domain A, then relation ~ defined by
x ~ y iff f(x) = f(y)
is an equivalence relation on A, and is called the kernel relation of f.

Equivalence class
Let R be an equivalence relation on a set S. if a ∈ S, then the equivalence class of a, denoted by [a], is the subset of S consisting of all elements that are equivalent to a. In other words, we have
[a] = { x ∈ S | x R a}

4.3 Order Relations

Definition of a partial order
A a binary relation is called a partial order if it is antisymmetric, transitive, and either reflexive or irreflexive.

Definition of a partially ordered set
The set over which a partial order is defined is called a partially ordered set, or poset. IF we want to emphasize the fact that R is the partial order that makes S a poset, we can write <S, R>.

Descending chains and minimality
If A is a well-founded set, then every nonempty subset of A has a minimal element. Conversely, if