Proving Theorems

Direct proof:

An implication $p \rightarrow q$ can be proved by showing that if p is true, then q is also true.

Example: Give a direct proof of the theorem "If n is odd, then n^2 is odd."

Idea: Assume that the hypothesis of this implication is true (n is odd). Then use rules of inference and known theorems to show that q must also be true (n^2 is odd).

Proving Theorems

n is odd.

Then n = 2k + 1, where k is an integer.

Consequently, $n^2 = (2k + 1)^2$.
$= 4k^2 + 4k + 1$
$= 2(2k^2 + 2k) + 1$

Since n^2 can be written in this form, it is odd.

Proving Theorems

Indirect proof:

An implication $p \rightarrow q$ is equivalent to its ***contra-positive*** $\neg q \rightarrow \neg p$. Therefore, we can prove $p \rightarrow q$ by showing that whenever q is false, then p is also false.

Example: Give an indirect proof of the theorem "If 3n + 2 is odd, then n is odd."

Idea: Assume that the conclusion of this implication is false (n is even). Then use rules of inference and known theorems to show that p must also be false (3n + 2 is even).

Proving Theorems

Prove: If 3n + 2 is odd, then n is odd.

Suppose n is even.
Then n = 2k, where k is an integer.

It follows that 3n + 2 = 3(2k) + 2
= 6k + 2
= 2(3k + 1)

Therefore, 3n + 2 is even.

We have shown that the contrapositive of the implication is true, so the implication itself is also true

… and now for something completely different…

Set Theory

(sections 2.1, 2.2)

Actually, you will see that logic and set theory are very closely related.

Set Theory

Set: Collection of objects ("elements")

$a \in A$	"a is an element of A" "a is a member of A"
$a \notin A$	"a is not an element of A"
$A = \{a_1, a_2, \ldots, a_n\}$	"A consists of a_1,…"

Order of elements is meaningless

It does not matter how often the same element is listed.

Set Equality

Sets A and B are equal if and only if they contain exactly the same elements.

Examples:

- $A = \{9, 2, 7, -3\}$, $B = \{7, 9, -3, 2\}$: $A = B$
- $A = \{\text{dog, cat, horse}\}$,
  $B = \{\text{cat, horse, squirrel, dog}\}$: $A \neq B$
- $A = \{\text{dog, cat, horse}\}$,
  $B = \{\text{cat, horse, dog, dog}\}$: $A = B$

Examples for Sets

“Standard” Sets:

Natural numbers $\mathbf{N} = \{0, 1, 2, 3, \dots\}$

Integers $\mathbf{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$

Positive Integers $\mathbf{Z}^+ = \{1, 2, 3, 4, \dots\}$

Real Numbers $\mathbf{R} = \{47.3, -12, \pi, \dots\}$
The description on the right is very misleading, since we can’t actually list all elements of **R** in a sequence.

Rational Numbers $\mathbf{Q} = \{1.5, 2.6, -3.8, 15, \dots\}$
(correct definition will follow)

Examples for Sets

$A = \varnothing$ “empty set/null set”

$A = \{z\}$ Note: $z \in A$, but $z \neq \{z\}$

$A = \{\{b, c\}, \{c, x, d\}\}$

$A = \{\{x, y\}\}$
Note: $\{x, y\} \in A$, but $\{x, y\} \neq \{\{x, y\}\}$

$A = \{x \mid P(x)\}$
“set of all x such that P(x)”

$A = \{x \mid x \in \mathbf{N} \wedge x > 7\} = \{8, 9, 10, \dots\}$
“set builder notation”

Examples for Sets

We are now able to define the set of rational numbers Q:

$\mathbf{Q} = \{a/b \mid a \in \mathbf{Z} \wedge b \in \mathbf{Z}^+\}$.
(We actually need equivalence classes of such pairs (a,b))

or $\mathbf{Q} = \{a/b \mid a \in \mathbf{Z} \wedge b \in \mathbf{Z} \wedge b \neq 0\}$

And how about the set of real numbers R?

$\mathbf{R} = \{r \mid r \text{ is a real number}\}$
That is the best we can do without getting much more sophisticated. A real variables book such as “Principles of Mathematical Analysis” by Walter Rudin will have the details, but that isn’t discrete math.

Subsets

$A \subseteq B$ “A is a subset of B”

$A \subseteq B$ if and only if every element of A is also an element of B.

Some people use $A \subset B$ to mean “A is a subset of B”.

We can completely formalize this:

$A \subseteq B \Leftrightarrow \forall x\, (x \in A \rightarrow x \in B)$

Examples:

$A = \{3, 9\}$, $B = \{5, 9, 1, 3\}$, $A \subseteq B$? true

$A = \{3, 3, 3, 9\}$, $B = \{5, 9, 1, 3\}$, $A \subseteq B$? true

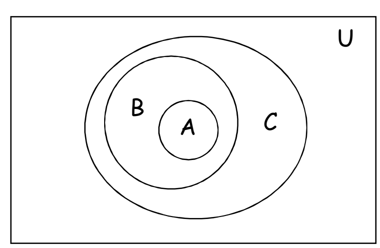
$A = \{1, 2, 3\}$, $B = \{2, 3, 4\}$, $A \subseteq B$? false

Subsets

Useful rules:

$A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$

$(A \subseteq B) \wedge (B \subseteq C) \Rightarrow A \subseteq C$ (see Venn Diagram)



Subsets

Useful rules:
$\emptyset \subseteq A$ for any set A
$A \subseteq A$ for any set A

Proper subsets:
$A \subset B$ "A is a proper subset of B"
$A \subset B \Leftrightarrow \forall x\ (x \in A \rightarrow x \in B) \wedge \exists x\ (x \in B \wedge x \notin A)$
or
$A \subset B \Leftrightarrow \forall x\ (x \in A \rightarrow x \in B) \wedge \neg\forall x\ (x \in B \rightarrow x \in A)$

Cardinality of Sets

If a set S contains exactly n distinct elements, $n \in \mathbf{N}$, we call S a finite set with cardinality n. $|S| = n$.

Examples:

A = {Mercedes, BMW, Porsche},	$\|A\| = 3$
$B = \{1, \{2, 3\}, \{4, 5\}, 6\}$	$\|B\| = 4$
$C = \emptyset$	$\|C\| = 0$
$D = \{ x \in \mathbf{N} \mid x \leq 7000 \}$	$\|D\| = 7001$
$E = \{ x \in \mathbf{N} \mid x \geq 7000 \}$	E is infinite!

The Power Set

2^A or $P(A)$ "power set of A"
$2^A = \{B \mid B \subseteq A\}$ (consists of all subsets of A)

Examples:

$A = \{x, y, z\}$
$2^A = \{\emptyset, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}$

$A = \emptyset$
$2^A = \{\emptyset\}$
Note: $|A| = 0$, $|2^A| = 1$

The Power Set

Cardinality of power sets:
$|\ 2^A\ | = 2^{|A|}$
Imagine each element in A has an "on/off" switch
Each possible switch configuration in A corresponds to one element in 2^A, namely the set of all elements that are "on".

A	1	2	3	4	5	6	7	8
x	x	x	x	x	x	x	x	x
y	y	y	y	y	y	y	y	y
z	z	z	z	z	z	z	z	z

- For 3 elements in A, there are $2 \times 2 \times 2 = 8$ elements in 2^A, that is, 8 subsets of A.

Cartesian Product

The *ordered n-tuple* $(a_1, a_2, a_3, \ldots, a_n)$ is an ordered collection of objects.

Two ordered n-tuples $(a_1, a_2, a_3, \ldots, a_n)$ and $(b_1, b_2, b_3, \ldots, b_n)$ are equal if and only if they contain exactly the same elements in the same order, i.e. $a_i = b_i$ for $1 \leq i \leq n$.

The *Cartesian product* of two sets is defined as:
$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$
Example: $A = \{x, y\}$, $B = \{a, b, c\}$
$A \times B = \{(x, a), (x, b), (x, c), (y, a), (y, b), (y, c)\}$

Cartesian Product

Note that:
$A \times \emptyset = \emptyset$
$\emptyset \times A = \emptyset$
For non-empty sets A and B: $A \neq B \Leftrightarrow A \times B \neq B \times A$
$|A \times B| = |A| \cdot |B|$

The Cartesian product of two or more sets is defined as:
$A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i \text{ for } 1 \leq i \leq n\}$

Set Operations

Union: $A \cup B = \{x \mid x \in A \vee x \in B\}$

Example: A = {a, b}, B = {b, c, d}
$A \cup B = \{a, b, c, d\}$

Intersection: $A \cap B = \{x \mid x \in A \wedge x \in B\}$

Example: A = {a, b}, B = {b, c, d}
$A \cap B = \{b\}$

Arbitrary unions and intersections

If S is some index set, finite or infinite, we define
$U_{(i \in S)} A_i = \{ x \mid x \in A_j \text{ for some } j \in S\}$
and
$\cap_{(i \in S)} A_i = \{ x \mid x \in A_j \text{ for all } j \in S\}$.

$U_{i=1}^{\infty} A_i = A_1 \cup A_2 \cup A_3 \cup A_4 \cup \ldots$
$\cap_{i=1}^{\infty} A_i = A_1 \cap A_2 \cap A_3 \cap A_4 \cap \ldots$
are special cases

Set Operations

Two sets are called disjoint if their intersection is empty, that is, they share no elements:
$A \cap B = \varnothing$

The difference between two sets A and B contains exactly those elements of A that are not in B:
$A\text{-}B = \{x \mid x \in A \wedge x \notin B\}$
Example: A = {a, b}, B = {b, c, d}, A-B = {a}

Set Operations

The *complement* of a set A contains exactly those elements in the universe of discourse that are not in A:
-A = U-A

Example: U = **N**, B = {250, 251, 252, …}
-B = {0, 1, 2, …, 248, 249}

Set Operations

Table 1 in Section 2.2 (page 130) shows many useful set identities.
How can we prove $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$?

Method I:
$x \in A \cup (B \cap C)$
$x \in A \vee x \in (B \cap C)$
$x \in A \vee (x \in B \wedge x \in C)$
$(x \in A \vee x \in B) \wedge (x \in A \vee x \in C)$
(distributive law for logical expressions)
$x \in (A \cup B) \wedge x \in (A \cup C)$
$x \in (A \cup B) \cap (A \cup C)$

Set Operations

Method II: Membership table
1 means “x is an element of this set”
0 means “x is not an element of this set”

A B C	$B \cap C$	$A \cup (B \cap C)$	$A \cup B$	$A \cup C$	$(A \cup B) \cap (A \cup C)$
0 0 0	0	0	0	0	0
0 0 1	0	0	0	1	0
0 1 0	0	0	1	0	0
0 1 1	1	1	1	1	1
1 0 0	0	1	1	1	1
1 0 1	0	1	1	1	1
1 1 0	0	1	1	1	1
1 1 1	1	1	1	1	1

Set Operations

Roughly speaking, every logical expression can be transformed into an equivalent expression in set theory and vice versa.

Exercises

Question 1:

Given a set A = {x, y, z} and a set B = {1, 2, 3, 4}, what is the value of $| 2^A \times 2^B |$?

Question 2:

Is it true for all sets A and B that $(A \times B) \cap (B \times A) = \varnothing$? Or do A and B have to meet certain conditions?

Question 3:

For any two sets A and B, if $A - B = \varnothing$ and $B - A = \varnothing$, can we conclude that A = B? Why or why not?

Functions

(section 2.3)

Functions

A *function* f from a set A to a set B is an assignment of exactly one element of B to each element of A.

We write

f(a) = b

if b is the unique element of B assigned by the function f to the element a of A.

If f is a function from A to B, we write

f: A→B and say “f maps A to B”

(note: Here, “→“ has nothing to do with if… then)

Functions

If f:A→B, we say that A is the *domain* of f and B is the *codomain* of f.

If f(a) = b, we say that b is the *image* of a and a is the *pre-image* of b.

The *range* of f:A→B is the set of all images of elements of A.

We say that f:A→B *maps* A to B.

Functions

Let us take a look at the function f:P→C with
P = {Linda, Max, Kathy, Peter}
C = {Boston, New York, Hong Kong, Moscow}

f(Linda) = Moscow
f(Max) = Boston
f(Kathy) = Hong Kong
f(Peter) = New York

Here, the range of f is C.

Functions

Let us re-specify f as follows:

f(Linda) = Moscow
f(Max) = Boston
f(Kathy) = Hong Kong
f(Peter) = Boston

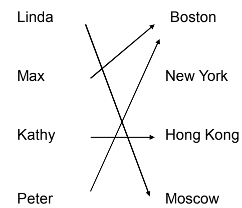
Is f still a function? yes

What is its range? {Moscow, Boston, Hong Kong}

Functions

Other ways to represent f:

x	f(x)
Linda	Moscow
Max	Boston
Kathy	Hong Kong
Peter	Boston



Functions

If the domain of our function f is large, it is convenient to specify f with a formula, e.g.:

f:**R**→**R**
f(x) = 2x

This leads to:
f(1) = 2
f(3) = 6
f(-3) = -6
…

Functions

Let f_1 and f_2 be functions from A to **R**.
Then the *sum* and the *product* of f_1 and f_2 are also functions from A to **R** defined by:
$(f_1 + f_2)(x) = f_1(x) + f_2(x)$
$(f_1f_2)(x) = f_1(x)\, f_2(x)$

Example:
$f_1(x) = 3x$, $f_2(x) = x + 5$
$(f_1 + f_2)(x) = f_1(x) + f_2(x) = 3x + x + 5 = 4x + 5$
$(f_1f_2)(x) = f_1(x)\, f_2(x) = 3x\,(x + 5) = 3x^2 + 15x$

Functions

We already know that the *range* of a function f:A→B is the set of all images of elements a∈A.

If we only consider a subset S⊆A, the set of all images of elements s∈S is called the *image* of S under f.

We denote the image of S by f(S):

f(S) = {f(s) | s∈S}

Functions

Let us look at the following well-known function:
f(Linda) = Moscow
f(Max) = Boston
f(Kathy) = Hong Kong
f(Peter) = Boston

What is the image of S = {Linda, Max} ?
f(S) = {Moscow, Boston}

What is the image of S = {Max, Peter} ?
f(S) = {Boston}

Properties of Functions

A function $f:A \to B$ is said to be *one-to-one* (or *injective*), if and only if

$\forall x, y \in A\ (f(x) = f(y) \to x = y)$

In other words: f is one-to-one (injective) if and only if it does not map two distinct elements of A onto the same element of B.

Properties of Functions

And again…

f(Linda) = Moscow

f(Max) = Boston

f(Kathy) = Hong Kong

f(Peter) = Boston

Is f one-to-one?

No, Max and Peter are mapped onto the same element of the image.

g(Linda) = Moscow
g(Max) = Boston
g(Kathy) = Hong Kong
g(Peter) = New York

Is g one-to-one?

Yes, each element is assigned a unique element of the image.

Properties of Functions

How can we prove that a function f is one-to-one?

Whenever you want to prove something, first take a look at the relevant definition(s):

$\forall x, y \in A\ (f(x) = f(y) \to x = y)$

Example:

$f:\mathbf{R} \to \mathbf{R}$

$f(x) = x^2$

Disproof by counterexample:

$f(3) = f(-3)$, but $3 \neq -3$, so f is not one-to-one.

Properties of Functions

… and yet another example:

$f:\mathbf{R} \to \mathbf{R}$

$f(x) = 3x$

One-to-one: $\forall x, y \in A\ (f(x) = f(y) \to x = y)$

To show: $f(x) \neq f(y)$ whenever $x \neq y$

$x \neq y$

$\Leftrightarrow 3x \neq 3y$

$\Leftrightarrow f(x) \neq f(y)$,

so if $x \neq y$, then $f(x) \neq f(y)$, that is, f is one-to-one.

The Growth of Functions: Big O

The growth of functions is usually described (for upper bounds) by using the **big-O notation**.

Definition: Let f and g be functions from the integers or the real numbers to the real numbers. We say that f(x) is O(g(x)) if there are constants C and k such that

$|f(x)| \leq C|g(x)|$ for all $x > k$.

(f is bounded above by g, up to a constant multiple. f grows no faster than g)

The Growth of Functions: Ω

The growth of functions is bounded below using the **Ω (capital Omega) notation**.

Definition: Let f and g be functions from the integers or the real numbers to the real numbers. We say that f(x) is *Ω(g(x))* if there are positive constants C and k such that

$|f(x)| \geq C|g(x)|$ for all $x > k$.

(f is bounded below by g, up to a constant multiple. f grows at least as fast as g)

The Growth of Functions: Θ

The growth of functions is also described using the **Θ (capital Theta) notation**.

Definition: Let f and g be functions from the integers or the real numbers to the real numbers. We say that f(x) is *Θ(g(x))* if there are positive constants C_1, C_2, and k such that

$C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$ for all $x > k$.

(f is bounded above and below by constant multiples of g: f grows at the same rate as g)

The Growth of Functions

When we analyze the growth of functions we generally consider f(x) and g(x) which are always positive.

In that case we can simplify the big-O requirement to

$f(x) \leq C \cdot g(x)$ whenever $x > k$.

If we want to show that f(x) is O(g(x)), we only need to find **one** pair (C, k) (which is never unique).

The Growth of Functions

The idea behind the big-O notation is to establish an **upper bound** for the growth of a function f(x) for large x.

This bound is specified by a function g(x) that is usually much **simpler** than f(x).

We accept the constant C in the requirement

$f(x) \leq C \cdot g(x)$ whenever $x > k$,

because **C does not grow with x.**

We are only interested in large x, so it is OK if $f(x) > C \cdot g(x)$ for $x \leq k$.

The Growth of Functions

Example:

Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$.

For $x > 1$ we have:

$x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2$

$\Rightarrow x^2 + 2x + 1 \leq 4x^2$

Therefore, for C = 4 and k = 1:

$f(x) \leq Cx^2$ whenever $x > k$.

$\Rightarrow$ f(x) is $O(x^2)$.

The Growth of Functions

Question: If f(x) is $O(x^2)$, is it also $O(x^3)$?

Yes. x^3 grows faster than x^2, so x^3 grows also faster than f(x).

Therefore, we always want to find the **smallest** simple function g(x) for which f(x) is O(g(x)).

The Growth of Functions

"Popular" functions g(n) are
n log n, 1, 2^n, n^2, n!, n, n^3, log n

Listed from slowest to fastest growth:

- 1
- log n
- n
- n log n
- n^2
- n^3
- 2^n
- n!

The Growth of Functions

A problem that can be solved with polynomial worst-case complexity is called ***tractable***.

Problems of higher complexity are called ***intractable.***

Problems that no algorithm can solve are called ***unsolvable***.

You will find out more about this in CS420.

Useful Rules for Big-O

For any **polynomial** $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$, where $a_0, a_1, \ldots, a_n$ are real numbers, $f(x)$ is $O(x^n)$.

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 + f_2)(x)$ is $O(\max(g_1(x), g_2(x)))$

If $f_1(x)$ is $O(g(x))$ and $f_2(x)$ is $O(g(x))$, then $(f_1 + f_2)(x)$ is $O(g(x))$.

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 f_2)(x)$ is $O(g_1(x)\, g_2(x))$.

Complexity Examples

What does the following algorithm compute?

procedure who_knows($a_1, a_2, \ldots, a_n$: integers)
m := 0
for i := 1 to n-1
    for j := i + 1 to n
        if $|a_i – a_j| > m$ **then** m := $|a_i – a_j|$
{m is the maximum difference between any two numbers in the input sequence}

Comparisons: n-1 + n-2 + n-3 + … + 1
$= (n – 1)n/2 = 0.5n^2 – 0.5n$

Time complexity is $O(n^2)$.

Complexity Examples

Another algorithm solving the same problem:

procedure max_diff($a_1, a_2, \ldots, a_n$: integers)
min := a_1
max := a_1
for i := 2 to n
    if a_i < min **then** min := a_i
    else if a_i > max **then** max := a_i
m := max - min

Comparisons: no more than 2n - 2

Time complexity is O(n).