CSCB36

Summary Of Vassos Hadzilacos'

Course notes for CSC B36/236/240

INTRODUCTION TO THE THEORY OF COMPUTATION

Computer & Mathematical Sciences
UNIVERSITY OF TORONTO
SCARBOROUGH

Instructor:	**Dr.Nick Cheng**
Email:	nick@utsc.utoronto.ca
Office:	IC348
Office Hours:	TBA

1 PRELIMINARIES

Sets: A collection of objects **(Elements)**.
If an object a is an element of set A, we say that a is a **member of** A; denoted $a \in A$
The collection that contains no elements is called the **empty** or **null** set' denoted $\varnothing$
Cardinality/Size: Number of elements in a set. The **cardinality** of set A is denoted $|A|$, and is a non-negative integer. If A has a infinite number of elements, $|A| = \infty$, and if $A = \varnothing$, then $|A| = 0$.

Extensional Description: Describing a set by listing its elements explicitly, e.g. $A = \{1, 4, 5, 6\}$
Intentional Description: Describing a set by stating a property that characterizes its elements, e.g. $A = \{x | \text{x is a positive integer less than 5}\}$

Let A and B be sets.
If every element of A is also an element of B,
then A is a **subset** of B ($A \subseteq B$), and B is a **superset** of A ($B \supseteq A$).

If $A \subseteq B$ and $B \subseteq A$, then A is **equal** to B ($A = B$).
If $A \subseteq B$ and $A \neq B$, then A is a **proper subset** of B; ($A \subset B$ and B) is a **proper superset** of A ($B \supset A$).
Note the empty set is a subset of every set, and a proper subset of every set other than itself.

The **union** of A and B ($A \cup B$), is the set of elements that belong to A or B (or both).
The **intersection** of A and B ($A \cap B$), is the set of elements that belong to both A and B.
If no elements belongs to both A and B, their intersection is empty, and they are **disjoint** sets.
The **difference** of A and B, ($A - B$), is the set of elements that belong to A but do not belong to B.
Note that: $A - B = \varnothing \iff A \subseteq B$
The **union** and **intersection** can also be defined for an arbitrary (even infinite) number of sets.
let I be a set of indices, such that for each $i \in I$ there is a set A_i

$$\cup_{i \in I} A_i = \{x : \text{for some } i \in I, x \in A_i\}$$
$$\cap_{i \in I} A_i = \{x : \text{for each } i \in I, x \in A_i\}$$

The **powerset** is the set of subsets, e.g. $A = \{a, b, c\}, \mathcal{P}(A) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
Partition of a set A, is pairwise disjoint subsets of A whose union is A. A partition of set A is a set $\mathcal{X} \subseteq \mathcal{P}(A)$, such that:

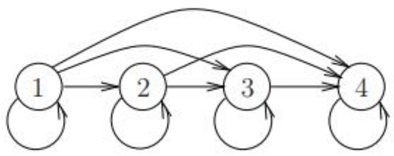
(i) for each $X \in \mathcal{X}, X \neq \varnothing$
(ii) for each $X, Y \in \mathcal{X}, X \neq Y$
(iii) $\cup_{X \in \mathcal{X}} X = A$

Ordered Pair: A mathematical construction that bundles two objects a, b together, in a particular order, denoted (a, b). By this definition, $(a, b) = (c, d) \iff a = c \wedge b = d$ and $(a, b) \neq (b, a)$ unless $a = b$.
We define an ordered pair (a, b) as the set $\{\{a\}, \{a, b\}\}$. We can also define ordered triples as ordered pairs, (a, b, c) can be defined as $(a, (b, c))$. This definition holds for ordered quadruples, quintuples, and ordered n-tuples for any integer $n > 1$.

Cartesian Product of A and B is denoted $A \times B$ and is the set of ordered pairs (a, b) where $a \in A, b \in B$.
$|A \times B| = |A| \cdot |B|$, note that if A, B are distinct nonempty sets, $A \times B \neq B \times A$
The Cartesian product of $n > 1$ sets $A_1, A_2, \ldots, A_n$ denoted $A_1 \times A_2 \times \cdots \times A_n$, is the set of ordered n-tuples $(a_1, a_2, \ldots, a_n)$, where $a_i \in A_i, i \in [1, n]$

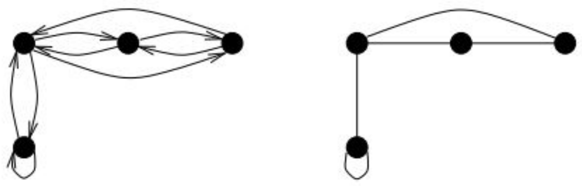
Relation R between sets A and B is a subset of the Cartesian product $A \times B$ ($R \subseteq A \times B$).
Arity: number of sets involved in the relation.
Two relations are **equal** if they contain exactly the same sets of ordered pairs. The two relations must refer to the exact same set of ordered pairs.

A binary relation (**arity** 2) between elements of the **same** set, can be represented graphically as a **directed graph**.
E.g. $R = \{(a, b)|a, b \in \{1, 2, 3, 4\}$ and $a \leq b\}$



R is a **reflexive** relation, if for each $a \in A, (a, a) \in R$, e.g. the relation $a \leq b$ between integers is reflexive, while $a < b$ is not.

R is a **symmetric** relation if for each $a, b \in R, (b, a) \in R$,
E.g. $R_1 = \{(a, b)|a$ and b are persons with atleast one parent in common$\}$ is symmetric. In the directed graph that represents a symmetric relation, whenever there is an arrow from a to b, there is an arrow from b to a. We can represent this with an **undirected graph**.



directed graph on the left, undirected graph on the right
R is a **transitive** relation if for each $a, b, c \in A, (a, b) \in R \wedge (b, c) \in R \longrightarrow (a, c) \in R$,
E.g. $R = \{(a, b)|a$ and b are persons and a is an ancestor of b$\}$
We see that if a is an ancestor of b, and b is an ancestor of c, then a is an ancestor of c.

R is an **equivalence relation** if it is reflexive, symmetric and transitive,
E.g. $R = \{(a, b)|a$ and b are persons with the same parents$\}$
a and a are the same person, thus have the same parents ($(a, a) \in R$), R is **reflexive**.
a and b share the same parents, b and a share the same parents ($(a, b) \in R, (b, a) \in R$), R is **symmetric**.
a and b share the same parents, b and c share the same parents, a and c must share the same parents. R is **transitive**.
thus we say R is an **equivalence relation**.

Let R be an equivalence relation and $a \in A$. The **equivalence class** of a under R is defined as the set $R_a = \{b|(a, b) \in R\}$, i.e., the set of all elements that are related to a by R.
If R is reflexive, then we know $\forall a \in A, R_a \neq \varnothing$
If R is transitive, then we know $\forall a, b \in R, R_a \neq R_b \longrightarrow R_a \cap R_b = \varnothing$

R is **partial order** if it is anti-symmetric and transitive.
R is **total order** if it is partial order and satisfies for each $a, b \in A$, either $(a, b) \in R$ or $(b, a) \in R$.

Let A and B be sets. A **function** f from A to B is a special kind of relation where each element $a \in A$ is associated with one element in B.
The relation $f \subseteq A \times B$ is a **function** if for each $a \in A$ there is exactly one $b \in B$ such that $(a, b) \in f$.

We write $f : A \to B$ to denote that f is a function from A to B, where A is the **domain** of the function, and B is the **range**.

The function $f : A \to B$ is:
onto/surjective if for every $b \in B$ there is at least one $a \in A$ such that $f(a) = b$
one-to-one/injective if for every element $b \in B$, there is at most one element $a \in A$ such that $f(a) = b$
bijective if it is **one-to-one** and **onto**, if $f : A \to B$ is a bijection, then $|A| = |B|$.
The **restriction** of a function $f : A \to B$ is to a subset A' of its domain, denoted $f|A'$, is a function $f' : A' \to B$ such that for every $a \in A, f'(a) = f(a)$.
An **initial segment** I of $\mathbb{N}$ is a subset of $\mathbb{N}$ with the following property: for any element $k \in I$, if $k > 0$ then $k - 1 \in I$. Thus, an initial segment of $\mathbb{N}$ is either the empty set, or the set $\{0, 1, 2, \ldots, k\}$ for some nonnegative integer k, or the entire set $\mathbb{N}$

let A be a set. A **sequence over** A is a function $\sigma : I \to A$, where I is an initial segment of $\mathbb{N}$.

Intuitively $\sigma(0)$ is the first element in the sequence, $\sigma(1)$ is the second and so on. if $I = \varnothing$, then σ is the **empty** or **null** sequence, denoted ϵ. if $I = \mathbb{N}$ then σ is an infinite sequence; otherwise it is a finite sequence. The **length** of σ is $|I|$ -i.e., the cardinality of I.

let $\sigma : I \to A$ and $\sigma' : I' \to A$ be sequences over the same set A, and suppose that σ is finite.
Informally, the **concatenation** of σ and σ', denoted $\sigma \circ \sigma'$ and sometimes as $(\sigma\sigma')$, is the sequence over A that is obtained by juxtaposing the elements of σ' after the elements of σ.
More precisely, if $I' = \mathbb{N}$ (i.e., σ is infinite), then if we let $j = \mathbb{N}$; otherwise let J be the initial segment $\{0, 1, \ldots, |I| + |I'| - 1\}$. Then $\sigma \circ \sigma'; J \to A$, where for any $i \in I, \sigma \circ \sigma'(i) = \sigma(i)$, and for any $i \in I, \sigma \circ \sigma(|I| + i) = \sigma'(i)$.
Informally, the **reversal** of σ, denoted σ^R, is the sequence of the elements of σ in revresed order, more precisely, $\sigma^R : I \to A$ is the sequence so that, for each $i \in I, \sigma^R(i) = \sigma(|I| - 1 - i)$.

Since, strictly speaking, a sequence is a function of a special kind, sequences $\sigma : I \to A$ and $\sigma' : I \to A$ is equal if and only if, for every $k \in I, \sigma(k) = \sigma'(k)$.

From the definitions of concatenation and equality of sequences, it is easy to verify the following facts:

1. For any sequences $\sigma, \epsilon \circ \sigma = \sigma$; if σ is finite, $\sigma \circ \epsilon = \sigma$.
2. For any sequences σ, τ over the same set, if is finite and $\circ\sigma = \sigma$, then $= \epsilon$; if σ is finite and $\sigma\circ = \sigma$, then $= \epsilon$.

A sequence σ is a **subsequence** of sequence if the elements of σ appear in and do so in the same order. for example $\langle b, c, f\rangle$ is a subsequence of $\langle a, b, c, d, e, f, g\rangle$. Note that we do not require elements of σ to be consecutive elements of , we only require that hey appear in the same order as they do in .

If, in fact, the elements of σ are consecutive elements of , we say that σ is a **contiguous subsequence** of τ.
Formally the definition of the subsequence relationship between sequences is as follows. let A be a set, I and J be initial segments of $\mathbb{N}$ such that $|I| \leq |J|$, and $\sigma : I \to A, J \to A$ be sequences over A. The sequence σ is a subsequence of if there is an increasing function $f : I \to J$ so that, for all $i \in I, \sigma(i) = (f(i))$. If σ is a subsequence of and is not equal to , we say that σ is a **proper subsequence** of .

An **alphabet** is a nonempty set Σ; the elements of an alphabet are called its **symbols**. A **string** (over alphabet Σ) is simply a finite sequence over Σ.
The empty sequence is a string and is denoted, as usual, ϵ. The set of all strings over alphabet Σ is denoted Σ^*. Note that $\epsilon \in \Sigma^*$, for any alphabet Σ.

Operations on strings and languages (From Lecture)
ill
$\Sigma = \{0,1\}$, A string over Σ is a finite sequence like $x = \langle 1,0,1,1\rangle$
Reversal: $x^R = (1011)^R = 1101$
Concatenation: $x^k = x \cdot x \ldots x$ where x is concatenated k times.
Equality: $x = y$ means $|x| = |y|$ and the ith element of x is equal to the ith element of y.
Substring: $y = a_1a_2\ldots a_i \ldots a_j \ldots a_n$ and $x = a_i \ldots a_j$ then x is a substring. by this definition, every string is a substring of itself.
Proper Substring: an x substring of y where $|x| < |y|$

A **language** over Σ is a nonempty subset of Σ^*
Recall that the set of all strings over alphabet Σ is denoted Σ^*.
A complementation: $\Sigma = \Sigma^* - L$, for a language L.
Union & Intersection: $L \cup L'$, $L \cap L'$
Concatenation: $L \cdot L' = LL = \{xy | x \in L, y \in L'\}$, $\varnothing \cdot L' = \varnothing$
Suppose $|L|, |L'|$ are finite, $|L \cdot L'| \leq |L| \cdot |L'|$. In general
(Kleene) Star: $L^* = \{x | x = \epsilon \vee x = y_1 \cdot y_2 \ldots y_k, k \geq 0, y_1, \ldots, y_k \in L\}$
For example $L = \{a,b\}, L* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, abb, \ldots\}$
Exponentiation: $L^k = L \cdot L \cdot L \cdot \cdots \cdot L$ L concatenated 10 times.
$L^k = \begin{cases} \{\epsilon\}, & k = 0 \\ \{L^{k-1} \cdot L, & k > 0 \end{cases}$, $L^* = \Cup_{k \in \mathbb{N}} L^k$
Reversal: (L^R) or $Rev(L) = \{x^R | x \in L\}$

Since strings are simply (finite) sequences over a specified set, various notions defined for sequences apply to string as well. In particular, this is the case for the notion of length which must now be a natural number, and cannot be inf. We use the term **substring** as synonymous to contiguous subsequence.

Let A be a finite set, A **permutation** of A is a sequence in which every element of A appears once and only once. For example if $A = \{a,b,c,d\}$ then $\langle b,a,c,d\rangle, \langle a,c,d,b\rangle \ldots$

Sometimes we speak of permutations of a sequence rather than a set. In this case, the definition is as follows: Let $\alpha : I \to A$ and $: I \to A$ be finite sequences over A. The sequence is a permutaiton of α is there is a bijective function $f : I \to I$ so that for every $i \in I, (i) = \alpha(f(i))$

2 INDUCTION

Fundamental properties of the natural numbers
The natural numbers are nonnegative integers $0, 1, \ldots$ denoted $\mathbb{N}$;
Principle of well-ordering: Any nonempty subset A of $\mathbb{N}$ contains a minimum element; i.e., $\forall A \subseteq N, \ni$: $A \neq \varnothing, \exists a \in A, \ni$: $\forall a' \in A, a \leq a'$
This applies to all nonempty subsets of $\mathbb{N}$ and to infinite subsets of $\mathbb{N}$. This principle does not apply to other sets.

Simple induction
Let A be any set that satisfies the following properties:
$0 \in A$
$\forall i \in \mathbb{N}, i \in A \Rightarrow i + 1 \in A$
Then A is a superset of $\mathbb{N}$.

Complete induction
Let A be any set that satisfies the following property:

$\forall i \in \mathbb{N}$, if evrey natural number less than $i \in A$ then $i \in A$.
Then A is a supserset of $\mathbb{N}$.
This principle is similar to the principle of simple induction, although there are some differences.
The requirement that $0 \in A$ is implicit as for any $i \in \mathbb{N}, 0 \leq \mathbb{N} \Rightarrow 0 \in A$. The second difference is we require i to be an element of A if all elements preceding i are in A. In contrast, we require $i \in A$ if $i - 1 \in A$.

Equivalence of the three principles
Theorem 1.1 The principle of well-ordering, induction, and complete induction are equivalent.
Proof. We prove this by establishing a "cycle" of implications. Specifically, we prove that (a) well-ordering implies induction, (b) induction implies complete induction, and (c) complete induction implies well ordering.
(a) well-ordering implies induction: Assume that the principle of well-ordering holds. We will prove that principle of induction is also true.
let A be a set that satisfies the following:
1. $0 \in A$
2. $\forall i \in \mathbb{N}, i \in A \Rightarrow i + 1 \in A$

We suppose that $A \not\supseteq \mathbb{N}$. Then there must exist a nonempty set $A' = \mathbb{N} - A$. By the principle of well ordering, A' has a minimum element a', by (1), $a' \neq 0$ because $0 \in A, a' \notin A$. Thus $a' > 0$ and $a' - 1$ is a natural number. But since a' is the minimum element of A', $a - 1 \notin A'$, and therefore $a' - 1 \in A$. By (2), as $a' - 1 \in A, a' \in A$ and we arrive at a contradiction, thus $A \supseteq \mathbb{N}$

(b) induction implies complete induction.
Again we try to prove that $A \supseteq \mathbb{N}$. To this end, define the set B,
$B = \{i \in \mathbb{N} |$ every natural number smaller than or equal to i is in $A\}$.
In other words, B is the initial segment of $\mathbb{N}$ up to but not including the smallest number that is not in A.
We see that $B \subseteq A$. Thus to prove that $A \supseteq \mathbb{N}$ it suffices to prove that $B \supseteq \mathbb{N}$
1. We have $0 \in B$ because (1) implies $0 \in \mathbb{A}$.
2. For any $i \in \mathbb{N}, i \in B \Rightarrow i + 1 \in B$.

These two above satisfy the conditions to use principle of induction (which we assume holds), we have $B \supseteq \mathbb{N}$, as wanted.

(c) Complete induction implies well-ordering: Assume that the principle of complete induction holds. Let A be any subset of $\mathbb{N}$ that has no minimum element. To prove that the principle of well-ordering holds, it suffices to show that A is empty.
For any $i \in \mathbb{N}$, if every natural number less than i is not in A, i is not in A either, otherwise, i would be the minimum element of A, and we are assuming that A has no minimum element. Let $A' = \mathbb{N} - A$. Thus any natural number i is in A if and only if i is not in A'.
Therefore, for any $i \in \mathbb{N}$, if every natural number less than i is in A', then i is also in A'. By the principle of complete induction, $A' \supseteq \mathbb{N}$. Therefore A is empty, as wanted. $\square$
Mathematical induction
Mathematical induction is a proof technique which can often be used to prove that a certain statement is true for all natural numbers.
For example:

1. $P_1(n)$: The sum of all natural numbers up to and including n is equal to $n(n + 1)/2$
2. $P_2(n)$: If a set A has n elements, then the powerset of A has 2^n elements.

$\vdots$

In general, a predicate of a natural number n may be true for all values of n, for some values of n, or for no values of n.

Examples of Induction:

Example 1. $S(n) : \displaystyle\sum_{t=0}^{n} t = \frac{n(n+1)}{2}$

BASIS: $n = 0$. In this case, $S(0)$ states that $0 = 0 \cdot 1/2$, which is obviously true. Thus $S(0)$

INDUCTION STEP: Let i be an arbitrary natural number and assume that $S(i)$ holds,

i.e., $\displaystyle\sum_{t=0}^{i} t = \frac{i(i+1)}{2}$. This is our inductive hypothesis.

WTS: $S(i+1)$. i.e., $\displaystyle\sum_{i=0}^{i+1} t = \frac{(i+1)(i+2)}{2}$

$$\begin{aligned}
\sum_{i=0}^{i+1} t &= \sum_{i=0}^{i} t + (i+1) \text{ [Definition of Addition]} \\
&= \frac{i(i+1)}{2} + (i+1) \text{ [IH]} \\
&= \frac{2(i+1) + i(i+1)}{2} \text{ [Algebra]} \\
&= \frac{2i + 2 + i^2 + i}{2} \text{ [Algebra]} \\
&= \frac{i^2 + 3i + 2}{2} = \frac{(i+1)(i+2)}{2} \text{ [Algebra]}
\end{aligned}$$

Therefore $S(i+1)$ as wanted. $\square$

Example 2. $P(k)$: any set of k elements has 2^k subsets.

BASIS: $k = 0$, then our set is $\{\}$ (the empty set), and as the empty set is a subset of itself, it is the only subset. We see that it indeed has $2^0 = 1$ subset(s). Thus, $P(0)$

INDUCTION STEP: Let i be an arbitrary natural number, assume that $P(i)$ holds, that is a set of i elements has exactly 2^i subsets. [IH]

WTS: $P(i+1)$. that is a set of $i+1$ elements has exactly 2^{i+1} subsets.

Consider an arbitrary set with $i+1$ elements, say $A = \{a_1, a_2, \ldots, a_{i+1}\}$. Consider two types of subsets of A : those that contain a_{i+1} and those that do not. let $\mathcal{Y}$ be the set of subsets of A that contain a_{i+1} and $\mathcal{N}$ be the set of subsets of A that do not.

First we see that $\mathcal{N}, \mathcal{Y}$ have the same number of elements. Next we see that they both have exactly i elements.

By the inductive hypothesis, $\mathcal{N}, \mathcal{Y}$ have exactly 2^i elements each.

Finally, we note that these sets of subsets are distinct, that is to say $\mathcal{N} \cap \mathcal{Y}\{\}$ as every subset in $\mathcal{N}$ will not contain a_{i+1} and every subset in $\mathcal{Y}$ will.

Since they are distinct, $|\mathcal{Y} \cup \mathcal{N}| = |\mathcal{Y}| + |\mathcal{N}| = 2^i + 2^i = 2 \cdot 2^i = 2^{i+1}$ as wanted. $\square$

Example 3. For any $m, n \in \mathbb{N}$, there are exactly n^m functions from any set of m elements to any set of n elements.

We prove this by using induction,

$P(m)$: for any $n \in \mathbb{N}$, there are exactly n^m functions from any set of m elements to any set of n elements or

$Q(n)$: for any $m \in \mathbb{N}$, there are exactly n^m functions from any set of m elements to any set of n elements

Proving either of these would give the desired proposition, but the first predicate is far easier.

BASIS: $m = 0$, $P(0)$ states that for any integer $n \in \mathbb{N}$, there are exactly n^0 functions from any set of m elements to any set of n elements. But since were consider functions that map zero elements to n elements, there is only one function whose domain is empty, the empty function.

Induction Step: Let i be an arbitrary natural number, suppose that for $P(i)$, i.e, there are n^i functions that map from any set of i elements to any set of n elements. [IH]

WTS: $P(i+1)$; i.e., there are n^{i+1} functions that map a set of $i+1$ elements to a set of n elements.
let A be an arbitrary set with $i+1$ elements, say $A = \{a_1, a_2, \ldots, a_{i+1}\}$, and B be an arbitrary set of n elements, say $B = \{b_1, b_2, \ldots, b_n\}$. Let's fix a particular element of a_{i+1}, and group the functions from $A \to B$ according to the element of B which a_{i+1} gets mapped.

$$\begin{aligned}
X_1 &= \text{the set of functions from } A \to B \text{ that map } a_{i+1} \text{ to } b_1\\
X_2 &= \text{the set of functions from } A \to B \text{ that map } a_{i+1} \text{ to } b_2\\
X_3 &= \text{the set of functions from } A \to B \text{ that map } a_{i+1} \text{ to } b_3\\
&\vdots\\
X_n &= \text{the set of functions from } A \to B \text{ that map } a_{i+1} \text{ to } b_n
\end{aligned}$$

Every functions from $A \to B$ belongs to one and only one of these sets. Notice that for any $j \in [1, n]$, X_j contains as many functions there are from the set $A' = \{a_1, a_2, \ldots, a_i\} \to B$. This is because X_j contains the functions from $\{a_1, a_2, \ldots, a_i, a_{i+1}\}$ to B where, however, we have specified the elemt to which a_{i+1} is mapped; the remaining elements of A can get mapped to the elements of B in any possible way, by induction hypothesis, $|X_j| = n^i$, for each $jm \ni: 1 \leq j \leq n$

$$|X_1| + |X_2| + \cdots + |X_n| = n^i + n^i + \cdots + n^i = n \cdot n^i = n^{i+1}, \square$$

Example 4. Let m, n be natural numbers such that $n \neq 0$. The division of m by n yields a quotient a remainder - i.e., unique natural numbers q, r such that $m = nq + r$ where $r < n$.
Define predicate $p(m)$: for any $n \in \mathbb{N}, n \neq 0, \exists q, r \in \mathbb{N}, \ni: m = q \cdot n + r$ where $r < n$.
Basis: $m = 0$, if we let $q = r = 0$ then for any $n \in \mathbb{N}, 0 = 0 \cdot n + 0$, thus $P(0)$
Inductive Step: Let $i \geq 0$ be an arbitrary natural number, and suppose $P(i)$; $\forall n \in \mathbb{N}, \exists q, r \in \mathbb{N}, \ni: i = nq + r, r < n$
WTS: $P(i+1)$ that is for any natural number $n \neq 0$, there are $q', r' \in \mathbb{N}, \ni: i+1 = q' \cdot n + r', r' < n$. Since $r < n$, we have two cases to consider, either $r < n-1$ or $r = n-1$

Case 1. $r < n-1$. In this case we have $i = nq + r$ by [IH], thus $i + 1 = nq + r + 1$
let $q' = q, r' = r + 1$, since $r < n - 1, r' = r + 1 < n$, we see that $i + 1 = q'n + r'$

Case 2. $r = n-1$. In this case again we have $i = nq + r$ by [IH], where $i + 1 = nq + r + 1$ however we have $r = n - 1$
Thus $i + 1 = nq + n - 1 + 1 = nq + n$, letting $q' = q + 1, r' = 0, i + 1 = nq' + r'$
We have shown that $P(i+1)$ holds for either case of r now we must show uniqueness.
Suppose that there exists $m, n \in \mathbb{N}, n \neq 0$ let $q, q', r, r' \in \mathbb{N}, \ni: m = q \cdot n + r = q' \cdot n + r', r, r' < n$
$(q - q') \cdot n = r' - r$, lets assume WLOG that $q \geq q' \Rightarrow q - q' > 0 \Rightarrow q - q' \geq 1$ which means that $(q - q') \cdot n \geq n$, but we see that $r - r' \geq n$ must hold which is impossible, since $r' < n$ and $r \geq 0$. Thus $q - q' = 0 = r - r'$
In other words, they are unique, as wanted.

Complete Induction
In carrying out the induction step of an inductive proof, we sometimes discover that to prove $P(i+1)$ holds, it would be helpful to know that $P(j)$ holds for more values of j that are smaller than $i+1$. For example, it might be that to prove $P(i+1)$ holds, we need to use the fact that $P(\lfloor i/2 \rfloor)$ holds and $P(i)$ holds.

Let $P(n)$ be a predicate of natural numbers, $\forall i \in \mathbb{N}, \forall j < i, P(j) \Rightarrow P(i)$

A typical proof of complete induction is as follows:

1. We let $i \in \mathbb{N}$
2. We assume that $P(j)$ for all natural numbers $j < i$ [IH]
3. Using the inductive hypothesis, we prove $P(i)$

Sometimes complete induction takes a different form, where it has explicit base cases. For example odd and even integers.

Bases other than zero

As with simple induction, sometimes we need to prove that a predicate holds, for all natural numbers greater than or equal to some constant c. We can prove such a statement by using variants of complete induction. For all $i \in \mathbb{N}, i \geq c$, if $P(j)$ holds forall $c \leq j < i$, then $P(i)$ holds.

Example 1.

$P(n)$: n has a prime factorization.

We use complete induction to prove that $P(n)$ holds for all natural $n \geq 2$. let $i \in \mathbb{N}$, $i \geq 2$ and assume $P(j)$ for any $2 \leq j < i$. WTS $P(i)$.

Case 1: i is a prime, then it is a prime factorization of itself, thus $P(i)$ for this case.

Case 2: i is not a prime, then there exists $d \in \mathbb{N}$ such that d divides i where $i = d \cdot a, a \in \mathbb{N}$ for $d \neq 1, d \neq i$, If $d \neq i$ and $d \in \mathbb{N}$, $d \geq 2$, the same is said about a, we see that $2 \leq a, b \leq i$. We can use our inductive hypothesis, and $P(a)$ as well as $P(b)$ holds, this implies that our i is a product of prime factors, thus $P(i)$.

Therefore, $P(n), \forall n \geq 2$

Example 2.

$P(n)$: Postage of exactly n cents can be made using only 4-cent and 7-cent stamps.

WTS, $P(n)\forall n \geq 18$m Let $i \geq 18 \in \mathbb{N}$ be an arbitrary, assume that $P(j)$ for any $18 \leq j < i$, WTS $P(i)$ holds.

Case 1: $18 \leq i \leq 21$. We can make postage for:

1. 18 cents using 1 4-cent stamp and 2 7-cent stamps.
2. 19 cents using 3 4-cent stamps and 1 7-cent stamp.
3. 20 cents using 5 4-cent stamps and 0 7-cent stamps.
4. 21 cents using 0 4-cent stamps and 3 7-cent stamps.

Case 2: $i \geq 22$ let $j = i - 4$ thus $18 \leq j < i$ and by inductive hypothesis, $P(j)$ holds, this means $j = 4 \cdot k + 7 \cdot l$. However if we let $k' = k + 1$ and $l' = l$, we see that $4 \cdot k' + 7 \cdot l' = 4 \cdot k + 7 \cdot l + 4 = j + 4 = i$

Thus $P(i)$, Thus in both cases, $P(i)$ as wanted.

Example 3. $P(n)$: if a full binary tree has n nodes, then n must be odd.

Let i be an arbitrary integer $i \geq 1$, suppose that $P(j)$ holds for all positive integers $j < i$. That is, for any positive integer $j < i$, if a full binary tree has j nodes, then j is an odd number. WTS $P(i)$, let T be an arbitrary binary tree with i nodes, WTS i is odd.

Case 1. $i = 1$, $P(i)$ for this case because T has only one node.

Case 2. $i > 1$. Let T_1 and T_2 be the two sub tress of T, and let i_1, i_2 be the number of nodes in them. The notes of T are: the roots, the nodes of T_1 and the nodes of T_2, thus $i = 1 + i_1 + i_2$

Since T is a full binary tree, T_1 and T_2 are also full binary trees. Since T has more than one node, each of T_1, T_2 has at least one node, so $i_1, i_2 \geq 1$.

But since i_1, i_2 are non negative numbers, $i = i_1 + i_2 + 1 \Rightarrow i \leq i_1, i_2 < i$; By the IH $P(i_1)$ and $P(i_2)$ holds. thus we have an even total.

3 CORRECTNESS

A **precondition** for a program is an assertion involving some of the variables of the program; this assertion states that what must be true before the program starts execution - in particular, it can describe what are deemed as acceptable inputs.
A **postcondition** for a program is an assertion involving some of the variables of the program; this assertion states what must be true when the program ends - in particular, it can describe what is a correct output.
Given a precondition/postcondition specification for a program, we say that the program is **correct** with respect to the specification, if whenever the precondition holds before the execution, then:

1. The program terminates.
2. When it does the postcondition holds.

Example 1.

4 7.2 Regular Expressions

Let Σ be an alphabet, Want to define the set regexes over Σ
- set of strings over Γ, where $\Sigma \cup \{(,),+,*,\epsilon,\varnothing\}$
Definition: Let $\mathcal{RE}$ let this be the smallest set. $\mathcal{RE}$ stands for regular expression.
Basis: $\epsilon, \varnothing; a \in \mathcal{RE}, \forall a \in \mathbb{Z}$
Induction Step: if $R, S \in \mathcal{RE}$, then $(RS), (R+S), R* \in \mathcal{RE}$
Syntax $\equiv$ Grammar, and Semanticsof regxes:
Given a regex R, wants to define the language of R $\mathcal{L}(R)$, i.e., if the set of strings that matches R
Define $\mathcal{L}(R)$ by structural induction.
Basis: $\mathcal{L}(\epsilon) = \{\epsilon\}$, $\mathcal{L}(\varnothing) = \varnothing, \mathcal{L}(a) = \{a\}, \forall a \in \Sigma$
If $R = (ST)$, then $\mathcal{L}(ST) = \mathcal{L}(S) \cdot \mathcal{L}(T)$
If $R = (S+T)$, then $\mathcal{L}(R) = \mathcal{S} \cup \mathcal{T}$
If $R = (s^*)$, then $\mathcal{L}(R) = \mathcal{L}(R)^*$
For example $1(1+0)^*$, $\{1\}, \{0\}$ by base cases of $\mathcal{L}(a) = \{a\}$
Thus $1+0 = \{0,1\}$, ofc $(1+0)^* = \{0,1\}*$
Finally $\{1\}\{0,1\}^*\{0\}$ is the set of all binary sequences starting from, ending with 0 with anything in between that is a binary sequence.
now $111^* \neq (111)^*$

Example
$\Sigma = \{0,1\}$
let $L = \{x \in \Sigma * | x \neq \epsilon, \text{ x begins and ends with the same symbol. }\}$
Find a regex $R, \ni$: $\mathcal{L}(R) = L, R = 0(0+1)*0 + 1(0+1)*1 + 0 + 1$
Justification the first expression matches all strings that start with 0, followed by any string of 0s and 1s, and ends with 0 ...
The + 0 + 1 at the end makes sure we have all strings that are of one length, as it matches the description of the set.

5 Section 7.3

Deterministic Finite State Automata (DFSA)
Definition: Formally a DFSA is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$, where:
Q is a finite set of states i.e., $q = \{q_1, q_2, \dots\}$
Σ is the input alphabet, i.e. M reads $\Sigma*$
δ is the transition function
s - initial state (starting state), $s \in Q$
F - the set of accepting states, $F \subseteq Q$
$\delta : Q \times \Sigma \to Q$, where $\delta(q,c) = q' \in Q$, where $q \in Q, c \in \Sigma$
when in state q and reading symbol c, go to state q'
The extended transition function
$\delta^* : Q \times \sigma^* \to Q$
$\delta^*(q,x) = q'$ means starting at state q, and reading string $s \in \Sigma*$, takes M to state q'
$$\delta^*(q,x) = \begin{cases} q & x = \epsilon \\ \delta(\delta^*(q,y),a) & x = ya \text{ where } y \in \Sigma^*, a \in \Sigma \end{cases}$$
M accepts x for $x \in \Sigma^*$ if and only if $\delta^*(s,x) \in F$

The Language of DFSA
$\mathcal{L}(M) = \{s \in \Sigma | \delta^*(s,x) \in F\}$ Formally or such that M accepts such s
Describe $\mathcal{L}(M)$, is the L that describes the language. to prove a $\mathcal{L}$

1. $\delta^*(q_0, x) = P(x) : \begin{cases} q_0 & x = \epsilon \\ q_1 & x \neq \epsilon \wedge \text{ x begins and ends with a 1} \\ q_2 & \text{“} \\ q_3 & x \neq \epsilon \wedge \text{ x begins with 1 and ends with 0} \end{cases}$

To Prove $P(x)$ holds for all $x \in \Sigma^*$ is very tedious, we will only need to figure out the State Invariant. Overlapping conditions cannot happen, where one string musn't satisfy multiple conditions.

6 Structural Induction for Regexes

Definition: A language $L \subseteq \Sigma^*$ is regular if there is a regex R such that $\mathcal{L}(R) = L$
Let $f : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$
Be a language operation, want to prove f preserves regular languages.
i.e., if L is regular, then $f(L)$
Predicate: $P(R)$: there's a regex $R', \ni: \mathcal{L}(R') = f(\mathcal{L}(R))$
Prove $P(R)$ holds for all regexes R by structural induction

Example: $\Sigma = \{0, 1\}$, define $f(L) = \{x \in \Sigma * | x \in L \wedge x$ ends with 0 $\}$
Structural Induction Proof
Basis:

1. If $R = \varnothing, f(\mathcal{L}(\varnothing)) = f(\varnothing) = \varnothing = \mathcal{L}(R')$ let $R' = \varnothing$
2. If $R = \epsilon, f(\mathcal{L}(\epsilon)) = f(\{\epsilon\}) = \varnothing = \mathcal{L}(R')$ let $R' = \varnothing$
3. If $R = 0, f(\mathcal{L}(0)) = f(\{0\}) = 0 = \mathcal{L}(R')$ let $R' = 0$
4. If $R = 1, f(\mathcal{L}(1)) = f(\{1\}) = \varnothing = \mathcal{L}(R')$ let $R' = \varnothing$

Induction Step: Let S, T be regexes. Suppose there are regexes:
$S', T' \ni: \mathcal{L}(S') = f(\mathcal{L}(S)), \mathcal{L}(T') = f(\mathcal{L}(T))$ i.e., $P(S), P(T)$ [Inductive Hypothesis]
WTP $P(R)$ for

1. $R = (S + T)$
   If $R = S + T$ then let $R' = S' + T'$
   $f(\mathcal{L}(S + T)) = f(\mathcal{L}(S) + \mathcal{L}(T)) = f((\mathcal{L}(S) \cup \mathcal{L}(T)) = f(\mathcal{L}(S)) \cup f(\mathcal{L}(T)) = \mathcal{L}(S') \cup \mathcal{L}(T')$
   $= \mathcal{L}(S' + T') = \mathcal{L}(R')$
2. $R = (ST)$
   If $R = ST$ then let $R' = \begin{cases} ST' + S' & \epsilon \in T \\ ST' & ow \end{cases}$
3. $R = S^*$ then let $R' = S^* S'$
   $f(\mathcal{L}())$

7 Finite State Automata and Regular Expressions

An **Alphabet** is a set Σ whose elements are called **symbols**. A **string** (over a specific alphabet Σ) is a finite sequence of symbols from Σ. The empty sequence is a string and denoted ϵ. The set of all strings over alphabet Σ is denoted Σ^*
The **concatenation** of strings x and y denoted $x \circ y$ or xy, is the sequence obtained by juxtaposing y after x; e.g., *abaa* $\circ$ *babb* is the string *abaababb*.
The **reversal** of a string x denoted $(x)^R$ is the string obtained by listing the elements of x in reverse order; e.g. $(abab)^R$ is the string *baba*.

For any $k \in \mathbb{N}$, we define the k-th **power** of a string x, denoted x^k by induction on k : $x^k = \begin{cases} \epsilon & k = 0 \\ x^{k-1} \circ x & k > 0 \end{cases}$
We see that exponentiation of strings is repeated concatenation.
Two string x, y are **equal**, denoted $x = y$, if $|x| = |y|$ and the ith symbol of x is the same as the ith symbol of y, for every $i \in [1, |x|]$.

A string x is a substring of string y if there exists strings x', x'' such that $x' \circ x \circ x'' = y$; if $x' \neq \epsilon$ or $x'' \neq \epsilon$ then x is a **proper substring** of y.
Theorem 7.4. For all string x and y, $(xy)^R = (y)^R(x)^R$
Proof. Let $P(y)$ be the predicate on strings. $P(y)$: for any string x, $(xy)^R = (y)^R(x)^R$
Basis: $y = \epsilon$. Then by the basis of the recursive definition of reversal $y = (y)^R = \epsilon$
$(xy)^R = (x\epsilon)^R = (x)^R = \epsilon(x)^R = (y)^R(x)^R$ as wanted.

Induction Step: Let y' be an arbitrary string such that $P(y')$ holds, i.e., for any $x, (xy')^R = (y')^R(x)^R$.
WTS, $\forall a \in \Sigma, y = y'a \Rightarrow P(y)$
$(xy)^R = (xy'a)^R = a(xy')^R = a((y')^R(x)^R) = (a(y')^R)(x)^R = (y'a)^R(x)^R = (y)^R(x)^R$ as wanted.

A **language** (over alphabet Σ) is a subset of $\Sigma*$. Note that a language may be an infinite set; each string in the language, however is finite. Also note that $\varnothing$ and $\{\epsilon\}$ are different languages.

Operations on Languages
let L, L' be languages over Σ. It is natural to combine these in order to obtain new languages.
Complementation: $\bar{L} = \Sigma^* - L$
Union: $L \cup L' = \{x | x \in L \vee x \in L'\}$
Intersection: $L \cap L' = \{x | x \in L \wedge x \in L'\}$
Concatenation: $L \circ L' = \{xy | x \in L \wedge y \in L'\}$
Kleene star: $L^\star$ Informally, the Kleene star is the set of all possible concatenations of 0 or more strings in L.

Formally it can be defined by induction as the smallest language such that:
Basis: $\epsilon \in L^\star$
Induction Step: If $x \in L^\star$ and $y \in L$ then $x \circ y \in L^\star$
Reversal: The reversal of L, $\textbf{Rev}(L) = \{(x)^R | x \in L\}$

Regular Expressions
Regular expressions are a precise and succinct notation for describing languages. An example of a regular expression is $(0 + 1)((01) * 0?)$
This regex describes the set of strings that:

1. Start with 0 or 1
2. Are then followed by zero or more repetitions of 01
3. and end with 0

Operators of regular expressions are:
Alternation: $+$, informally meaning ”or”
Concatenation: informally meaning ”followed by”
Repetition: $*$ informally meaning ”zero or more repetitions of”. The parentheses are used to indicate the order of operations.

let Σ be a finite alphabet. The set of **regular expressions** $\mathcal{RE}$ (over Σ) is the smallest set such that:
BASIS: $\varnothing, \epsilon$ and a (for each $a \in \Sigma$) belong to $\mathcal{RE}$
INDUCTION STEP: if R and S belong to $\mathcal{RE}$, then $(R+S)$, (RS) and $(R)^*$ belong to $\mathcal{RE}$
Each regular expression denotes a language over Σ.
(0) denotes the language $\{0\}$
$(0+1)$ denotes the language $\{0,1\}$
$(11)^*$ denotes the language $\{\epsilon, 11, 1111, \dots\}$
$(((01)+(10))(11)^*)^*$ denotes the language consisting of strings that are the concatenation of zero or more strings, each of which starts with either 01 or 10 then it is followed by zero or more repetitions of the string 11.

$\mathcal{L}(R)$ is the language denoted by a regular expression R, defined by structural induction on R:
BASIS: If R is a regular expression, then either $R = \varnothing, R = \epsilon$ or $R = a$ for some $a \in \Sigma$, For each of these cases we define $\mathcal{L}(R)$:

1. $\mathcal{L}(\varnothing) = \varnothing$ (the empty language, consisting of no strings)
2. $\mathcal{L}(\epsilon) = \{\epsilon\}$ (the language consisting of just empty strings)
3. $\forall a \in \Sigma, \mathcal{L}(a) = \{a\}$ (the language consisting of just the one-symbol string a).

INDUCTION STEP: If R is a regular expression, then $R = (S+T)$, $R = (ST)$ or $R = S^*$ For each of the three cases we define $\mathcal{L}(R)$:

1. $\mathcal{L}((S+T)) = \mathcal{L}(S) \cup \mathcal{L}(T)$
2. $\mathcal{L}((ST)) = \mathcal{L}(S) \circ \mathcal{L}(T)$
3. $\mathcal{L}(S^*) = (\mathcal{L}(S))^*$

Precedence rules
Concatenation always comes before union. $RS^* + T \equiv ((RS^*) + T)$
Examples
Let $\Sigma = \{0,1\}$

1. let L_1 be the set of strings in Σ^* that contains at most two 0s.

   A regex for L_1 is $R_1 = 1^* + 1^*01^* + 1^*01^*01^*$. We see that another regular expression that denotes this is $R_1^{'} = 1^*(0+\epsilon)1^*(0+\epsilon)1^*$

2. Let L_2 be the set of strings that contain an even number of 0s.

   A regex for L_2 is $R_2 = (1^*01^*01^*)^*$ or $R_2^{'} = 1^*(01^*1^*)^*$

   Now we need to prove that $\forall x \in L_2 \Leftrightarrow x \in \mathcal{L}(R_2^{'})$

   [IF] Let x be an arbitrary string in $\mathcal{L}(1^*(01^*1^*)^*)$. Thus, there is some $k \in \mathbb{N}$ and $y_0, y_1, \dots, y_k \in \Sigma^*$ such that $y_0 \in \mathcal{L}(1^*)$ and $y_i \in \mathcal{L}(01^*01^*)$ for $i \in [1,k]$

   therefore we have $l_0, l_1, \dots, l_k \in \mathbb{N}$ and $m_1, \dots, m_k \in \mathbb{N}$ such that

   $y_0 = 1^{l_0}$ and $y_i = 01^{l_i}01^{m_i}$ for $i \in [1,k]$

   Thus we see that x has exactly $2k$ 0s.

Formally **DFSA** is a 5-tuple M, where $M = (Q, \Sigma, \delta, s, F)$
Q - the **finite** set of states
Σ - the alphabet that the DFSA M reads
δ - the transition function
s - the starting (initial) state, $s \in Q$
F - the set of accepting states, $F \subseteq Q$

now $\delta : Q \times \Sigma \to Q$