Lecture 12: Relations

**Cartesian Product –** of the sets *S* and *T*, denoted by *S* x *T*,is the set of all ordered pairs (*s*, *t*) where *s Є S* and *t Є T*.

Note: (*s*, *t*) (*t*, *s*)

**Relation –** A structure used for representation of relationships.

Let *A* and *B* be two sets. A (binary) relation from *A* to *B* is a subset of A x B.

**Relation Inverses –** The inverse of a relation from *X* to *Y* is a relation from *Y* to *X*.

Relations can be represented in a Boolean, zero-one, matrix. A one represents a relation, while a zero represents no relation.

**Reflexive** – A relation *R* on a set *X* is called **reflexive**if *xRx* for every element *x* of *X*. The element has a relation with itself.

**Symetric –** A relation *R* on a set *X* is called **symmetric** if *xRy* 🡪 *yRx* for all *x*,*y* Є *X.*

***Antisymetric –*** A relation *R* on a set *X* is called **antisymetric** if

*xRy* ^ *xy* 🡪 *y~~R~~x* for all *x*,*y* Є *X*.

**Transitive -** A relation *R* on a set *X* is called **transitive** if

*xRy* ^ *yRz* 🡪 *xRz* for all *x*,*y,z* Є *X.*

Lecture 10: 2nd Order Linear (cont.)

Unless *b* = *c* = 0, the recurrence *axn*+ *bxn-1* + *cxn-2* = 0 has infinitely many solutions (sequences, called the **general solution** to the recurrence. All of them can be obtained by a single formula; the type of this formula depends on the roots of the quadratic equation called

the **characteristic equation** for the above recurrence: *ar2*+ *br* + *c* = 0.

**2 Theorems for Particular Solution -**

**1)** If the characteristic equation has two distinct real roots, *r*1, *r*2,

then *xn* = *c*1*r*1*n* + *c*2*r*2*n* where *c*1 and *c*2 are any two real

numbers.

**2)** If the characteristic equation has two equal real roots,

*r*1 = *r*2 = r then *xn* = *c*1*rn* + *c*2*rn* where *c*1 and *c*2 are any two real

numbers.

*c*1 and *c*2 will be provided when asked to determine the particular solution, so simply substitute the two coefficients and solve for *xn*.

Lecture 11: Matrices

***m* x *n* Matrix –** is a rectangular array (table) of numbers with *m* rows and *n* columns. If *m* = *n*, the matrix is said to be a **square**. Matrices are considered to be equal if they have the same dimensions and the same element in each corresponding location.

**Matrix Addition** can only occur when the matrices to be added have the same dimensions. Simply add elements from corresponding location. Resulting matrix will have the same dimensions as the original.

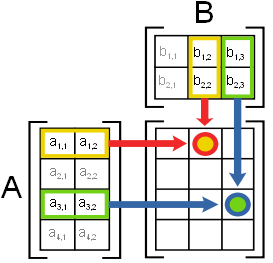
**Matrix Multiplication** **–**

**By a constant:** Simply multiply each element by the constant.

**By a Matrix:** Let Matrix *A* be *n* x *k* and Matrix *B* be *k* x *m*. Matrix

*C* (result) will be an *n* x *m* matrix. If these conditions are not

satisfied, multiplication is not possible. Multiply as follows:



Matrix multiplication is not commutative.

**Identity Matrix of order *n* –** the *n* x *n* matrix *In* = [*aij*]

where *aij* = 1 if i = j or *aij* = 0 if ij.

**Matrix Inverse –** Let *A* = [*aij*] be an *n* x *n* matrix. The **inverse** of *A*, denoted by *A*-1, is the *n* x *n* matrix such that *AA*-1 = *A*-1*A* = *In.* If this matrix exists, *A* is called **invertible** or **nonsingular**, otherwise *A* is called **singular**.

**Matrix Transposition** **–** Rows simply become columns.

**Matrix Symetry –** If transposed matrix is equal to original, they are said to be **Symetric**.

**Matrices Join and Meet –** These properties hold true to disjunction (join) and conjunction (meet) properties for **zero-one matrices,** those which are filled with 0’s and 1’s representing false and true binary values respectively.

**Boolean Product** – Matrix Multiplication can be applied to zero-one matrices with the following formula:

*cij* = (*ai*1 ^ *b*1*j*) v (*ai2* ^ *b2j*) v … v (*ai*k ^ *b*k*j*)

Lecture 10: 2nd Order Linear

**2nd Order Linear Homogeneous Recurrence with Constant Coefficients –** Recurrence that can be written in the form: *axn*+ *bxn-1* + *cxn-2* = 0 where *a*, *b*, and *c* are real and *a* .

Lecture 9: Recurrence Relations

**Recurrence Relation –** An equation that defines the generic term of a sequence *sn* in terms of one or more of its predecessors. An **initial condition** must be defined in order to define a recurrence relation.

**Solving –** To solve a recurrence relation subject to a given initial condition means to find a formula expressing its generic term as a function of *n*, the index of the sequence.

**Method of Forward Substitutions –**

**-** Start with the initial condition and generate a few

first terms in the hopes of seeing a pattern.

**-** Express the pattern by a formula.

**-** Prove the formula’s validity. (Substitute in both the

recurrence **and** the initial condition.

**Method of Backward Substitutions (Preferred) –**

**-** Using the recurrence, generate a few terms

preceding *xn* in a hope to see a pattern.

**-** Express the pattern by a formula.

**-** Consider the pattern’s case that corresponds to the

initial condition to take advantage of the latter.

**-** Prove the formula’s validity. (Substitute in both the

recurrence **and** the initial condition.

Lecture 8: Sequences and Sums (cont.)

**Generic Term Formula for Arithmetical Progression**:

*sn* = *a* + *nd* for *n* > 0 or*sn* = *a*+(*n*-1)*d* for *n* > 1

**Geometric Progression –** a numeric sequence that starts with some **initial term**, *a*, and obtains any other term by multiplying the term’s immediate predecessor by the same number, *r,* the **common ratio**.

Example: a, ar, ar2, ar3, …

**Generic Term Formula for Geometric Progression:**

*sn = arn* for *n* > 0 or *sn = arn-1* for *n* > 1

**Fibonacci Numbers –** a numeric sequence that starts with 0 and 1 and obtains any other term by adding the term’s two immediate predecessors.

**Generic Term Formula for Fibonacci Numbers:**

*Fn* = *qn /* rounded to nearest integer

where q = (1 + ) / 2 1.618

**Summation or Sigma Notation -** i = *a1 + a2 + … + an*

**Summation Formulas –**

More Generally

More Generally

**Summation Rules –**

Lecture 5: Strong Form of Math. Induction

**Strong Form of Mathematical Induction –** Basis Step: *P(n(0))* is true and Inductive Step: For every *k* > *n(0)* if *P(j)* is true for all *n(0)* < *j* < *k*, then *P(k+1)* is true.

Lecture 6: Sets

**Set –** A collection of distinct or discrete objects considered as a whole.

**Subset (Є)** **–** A set of elements contained by another set. Every set has at least one subset, an empty set (Ø). Every set is a subset of itself.

**Power Set –** A set of all subsets of an original set. If an original set has *n* elements, then its power set contains *2n* elements.

**Union (U) –** An operation which widens a set contents to include all elements of effected sets with no duplicates.

**Intersection (Π) –** An operation which restricts a set’s contents to include only elements included in all effected sets with no duplicates.

**Difference (-) –** An operation which restricts a set’s contents to eliminate elements only contained in all effected sets.

**De Morgan’s Law for Sets –** A U B == A Π B and A Π B == A U B

Lecture 7: Functions

**Function (*f*) –** from *A* to *B* is an assignment of exactly one element of *B* to each element of *A* where *A* and *B* are two nonempty sets. Can be determined by a vertical line test.

***f:* *A*🡪*B* or *f* maps *A* to *B* –** Denotes that *f* is a function from *A* to *B*.

**Domain –** *A* is the domain, origin or pre-image of *f*.

**Codomain –** *B* is the codomain, range or image of *f*.

**Injective or One to One Function** – No element of the codomain is the image of two or more distinct elements of the domain. Strictly increasing or decreasing functions are always considered One to One.

**Surjective Function –** Every element of the codomain is the image of at least one element of the domain.

**Bijective Function –** A function which is both Injective and Surjective.

**Inverse Function (f-1) *–*** The function from B to A that assigns to every element *b* of *B* the (unique) element *a* of *A* such that *f(a)* = *b* where *f* is a One to One Correspondence from *A* to *B*. Can be determined by a horizontal line test. Only bijective functions are invertible.

**Composition of Functions –** Denoted by *f ₀ g* and *g ₀ f* and defined as *f(g(x))* and *g(f(x))* respectively.

**Graph of a Function** **–** Defined by the ordered pairs of the related elements from it’s image and pre-image and plotted on the Cartesian Plane.

Lecture 8: Sequences and Sums

**Sequence** **–** An ordered list of items, called **terms**. More formally, a function whose domain is a set of consecutive integers (usually beginning with 0 or 1). If all terms of a sequence are numbers, it’s called a **numeric**. Sequences can also be defined as **finite** or **infinite**. They are defined by a formula for a generic term, *sn*, involving index *n*.

Example: 2, 4, 6, 8 … *sn = 2n* for *n* > 1

Or by an equation relating the generic term, *sn*, to one or more preceding terms of the sequence.

Example: 2, 4, 6, 8 … *sn = sn-1* + 2 for *n* > 1, *s1* = 2

**Arithmetical Progression –** a numeric sequence that starts with some **initial term**, *a,* and obtains any other terms by adding to the term’s immediate predecessor the same number, *d*, the **common difference**.

Example: *a*, *a*+*d*, *a*+2*d*, *a*+3*d*, …

Lecture 4: Mathematical Induction

**Mathematical Induction –** A method of proving propositions of the form “For every *n* > *n(0)*, *P(n)*” where *n(0)* is some integer (often 0 or 1) by proving Basis Step: *P(n(0))* is true, and Inductive Step: If *P(k)* is true, then *P(k+1)* is true for every *k* > *n(0)*.

**L-trominoes –** Example of Recursive Algorithm

Lecture 3: Methods of Proof

**Axiom (postulate) –** Underlying assumption, does not require a proof.

**Rules of Inference –** Used to draw conclusions from other assertions.

**Proof of a statement *A* –** A sequence of statements, which each

- is an axiom (postulate).

- or follows from earlier statements.

- and last statement is *A*

**Informal vs. Formal Proof –** Uses rules of interference informally and formally respectively.

**Theorem –** A statement which can be shown to be true.

**Lemma –** A theorem used in the proof of other theorems.

**Corollary –** A theorem that immediately follows from another theorem.

**Conjecture –** A likely-to-be-true statement yet with no proof found.

**Direct Proof –** Proof of implication *p* → *q* by showing that if *p* is true then *q* is also true, based directly on axioms, definitions, and previously proved theorems.

**Indirect Proof (by Contrapositive) –** Proof based on proving an assertion given in the form *p* → *q* by proving its contrapositive -*p* → -*q*, which is logically equivalent to it.

**Proof by Cases –** Proof based on partitioning the theorem’s domain into subdomains and proving the theorem separately for each of these subdomains.

**Proof by Contradiction –** Done by assuming that the conclusion *q* is false and showing that this assumption leads to a contradiction (given the hypothesis *p* is true).

**Proof of Equivalence –** Is denoted by an expression “*p* ↔ *q*”, “if and only if”, or “necessary and sufficient” and requires two separate proofs. *p* → *q* and *q* → *p*.

**Proof by Examples** – Proof by providing specific examples does not constitute a proof unless the examples include all from the domain. Disproof, however, requires only one specific counterexample.

Lecture 2: Predicates and Quantifiers

**Propositional Function *P(x)* –** A function whose values are propositions, i.e., it is an assignment to each element x of the function’s domain D called the *domain of discourse* a proposition (a true or false statement).

**Universal Quantification of *P(x)* –** “*P(x)* is true for all values of *x* in its universe of discourse.”

“for all *x* *P(x*)”

“for every *x* *P(x)*”

“*x* *P(x)*”

**Existential Quantification of *P(x)* –** “There exists an element *x* in its universe of discourse such that *P(x)* is true.”

“there is an *x* such that *P(x)*”

“for some *x* *P(x)*”

“*x P(x)*”

**Generalized De Morgan’s Law - -**(*x* *P(x))* == *x* – *P(x)*

Example: “Not every student has a computer” is equivalent

to “There is a student who doesn’t have a computer”

Lecture 1: Discrete Structures

**Discrete –** Consisting of discrete or unconnected elements.

**Discrete Structures –** Abstract mathematical structures used to represent discrete objects and relationships among them.

**Proposition –** A statement that is either true (1) or false (0)

|  |  |  |
| --- | --- | --- |
| **Negation (-)** | | |
| **p** | | **-p** |
| T | | F |
| F | | T |
| **Conjunction (^)** | | |
| **p** | **q** | **p ^ q** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |
| **Disjunction (v)** | | |
| **p** | **q** | **p v q** |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |
| **Exclusive Or (Ѳ)** | | |
| **p** | **q** | **p Ѳ q** |
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |
| **Implication (→)** | | |
| **p** | **q** | **p → q** |
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |
| **Biconditional (↔)** | | |
| **p** | **q** | **p ↔ q** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

|  |  |
| --- | --- |
| **Original** | **p → q** |
| **Converse** | **q → p** |
| **Inverse** | **-p → -q** |
| **Contrapositive** | **-q → -p** |

|  |  |
| --- | --- |
| **1st De Morgan’s Law** | **- (p v q) == (-p) ^ (-q)** |
| **2nd De Morgan’s Law** | **- (p ^ q)** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1st De Morgan’s Law Proof** | | | | | | |
| **p** | **q** | **(p v q)** | **-(p v q)** | **-p** | **-q** | **-p ^ -q** |
| T | T | T | F | F | F | F |
| T | F | T | F | F | T | F |
| F | T | T | F | T | F | F |
| F | F | F | T | T | T | T |