APPLIED ALGEBRA (DIVISION INTO SECTIONS APPROXIMATE ONLY)

§1 SET THEORY

① NOTATION

$\Rightarrow$	implies
$\Leftrightarrow$	if and only if
$\exists$	there exists
$\forall$	for all
$)$	such that
$\wedge$	and
$\vee$	or
$\neg$	not

} LOGICAL NOTATION

$\in$	element of
$\cup$	union with
$\cap$	intersection with
$\subseteq$	is contained in (subset of)
$\subset$	is contained in, but not equal to (proper subset)
$=$	equivalence
$\#$	number of elements in

} SET NOTATION

② POWER SETS

2^A denote the set of subsets of A, called exponentiating A.
The set of all sets is 2^U, where U is the universal set.

PROVING SET IDENTITIES

Three methods are available:

1) Elements: Let $x \in LHS$
also that $x \in RHS$.

2) Venn diagrams - number sectors and use these to prove.

(2) 3) Axiomatic method using identities.

THE SET LAWS

($\cup$ and $\cap$ can be interchanged, but we must $\emptyset$ and U in that case)

1) Commutativity $A \cup B = B \cup A$

2) Associativity $A \cup (B \cup C) = (A \cup B) \cup C$

3) Distributivity $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

4) Identity $A \cup \emptyset = A$

5) Complement $A \cup A' = U$

6) Involution $(A')' = A$

7) Idempotent $A \cup A = A$

8) Null $A \cup U = U$

9) Absorption $A \cup (A \cap B) = A$

10) De Morgan $(A \cup B)' = (A') \cap (B')$

Note for any set A, $\emptyset \subset A \subset U$

INDEXED SETS

Given a collection of sets A_α such that α is the index of each set, then this collection is said to be indexed by K if K is the set of all α.

UNION $\bigcup_{\alpha \in K} A_\alpha = \{ x \mid \exists \alpha \ni x \in A_\alpha \}$

$= \{ x \mid x \in A_\alpha \text{ for some } \alpha \in K \}$

(3) INTERSECTION $\bigcap_{\alpha \in K} A_\alpha = \{ x \mid \forall \alpha \in K \Rightarrow x \in A_\alpha \}$

$= \{ x \mid x \in A_\alpha \text{ for each } \alpha \in K \}$

We can also write $\bigcup_{\alpha \in K} A_\alpha = \bigcup_{i=1}^{m} A_i$ and $\bigcap_{\alpha \in K} A_\alpha = \bigcap_{i=1}^{m} A_i$

for $K = \{ 1, 2, 3, \ldots, m \}$

§2 RELATIONS

(4) RELATIONS BETWEEN SETS

SET PRODUCTS (CARTESIAN PRODUCT) $A \times B = \{(a,b) \mid a \in A \wedge b \in B\}$

$A^n = \underbrace{A \times A \times \dots \times A}_{n\text{-times}}$. Note $\#(A \times B) = \#(A)\,\#(B)$ for finite sets

[11] UNARY RELATIONS

A unary relation over a set A is a subset of A, usually conceived of as a property of some elements of A

(5) BINARY RELATIONS

A subset ρ of $A \times A$ is called a relation on A. We write $a \rho b \iff (a,b) \in \rho$ and say that a is related to b under ρ.
The converse of ρ is $\breve{\rho} = \{(a,b) \mid (b,a) \in \rho\}$

(6) RELATION MATRICES

Let $A = \{a_1, a_2, a_3, \dots, a_n\}$
$B = \{b_1, b_2, b_3, \dots, b_m\}$

Let $\rho \subset A \times B$ form the matrix denoted $E(\rho)$

$$E(\rho) = \begin{array}{c} a_1 \\ a_2 \\ \vdots \\ a_n \\ \\ b_1 \end{array} \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ & & & \\ c_{n1} & \dots & & c_{nm} \end{pmatrix} \begin{array}{c} \\ \\ \\ \\ \\ b_2 \quad \dots \quad b_m \end{array}$$

The c_{ij} is either a zero (if $(a_i, b_j) \notin \rho$) or a one (if $(a_i, b_j) \in \rho$, i.e. $a_i \rho b_j$)

(6) GRAPH OF RELATION

Let ρ be a relation on A. The graph of ρ, denoted $G(\rho)$, consists of the elements of A called nodes and arrows indicating the related elements (i.e. an arrow is equivalent to a 1 in the relation matrix)

(7) COMPOSITION OF RELATIONS

Let $\rho_1 \subset A \times B$

$\rho_2 \subset B \times C$

Then $\rho_1 \rho_2 = \rho_1 \circ \rho_2 = \{(a,c) \subset A \times C \mid \exists b \Rightarrow (a,b) \in \rho_1 \wedge (b,c) \in \rho_2\}$

If ρ is a relation on A, we can compose it with itself m times, where we write $\rho^m = \rho \circ \rho \circ \cdots \circ \rho$ (m terms)

(8) PRODUCTS OF RELATION MATRICES

Let $E(\rho_1)$ and $E(\rho_2)$ be rel mats. $E(\rho_1) \cdot E(\rho_2)$ is found by multiplying the matrices in the usual way, but subject to the following definitions: $1 + 1 = 0 + 1 = 1 + 0 = 1$

The set $\{0, 1\}$ with multiplication (and) and addition (or) subject to this definition and $1' = 0$ and $0' = 1$ is called the 2-element Boolean algebra.

PATHS OF LENGTH n

In a graph of a relation, a path of n arrows is called a path of length n. A graph $n \cdot G$ is the graph with the same vertices as G but with arrows indicating vertices related by paths of length n.

Note that $E(R_1 \cdot R_2) = E(R_1) \cdot E(R_2)$

and $G(R^2) = 2 \cdot G(R)$

(11) Similarly $E(R^n) = \underbrace{E(R) \cdot E(R) \cdots E(R)}_{n \text{ times}}$

and $G(\wp^n) = n \cdot G(\wp)$

and $a \wp^n b \Leftrightarrow a \wp a_1 \wedge a_1 \wp a_2 \wedge \ldots \wedge a_{n-1} \wp b$

for some $a_1, a_2, \ldots, a_{n-1}$

TYPES OF RELATION

REFLECTIVE : $a \wp a$ for all $a \in A$

in graph, each vertex must have ↺

in matrix, diagonal must be all 1's.

SYMMETRIC : $a \wp b \Leftrightarrow b \wp a$

(12) in graph : ○ ⇄ ○

in matrix must be symmetric about diagonal

TRANSITIVE : $a \wp b \wedge b \wp c \Rightarrow a \wp c$

in graph : ○ → ○ → ○ , ○ → ○

(14) TRANSITIVE CLOSURE OF ρ

Let ρ_1 and ρ_2 be two relations on A

Then $E(\rho_1 \cup \rho_2) = E(\rho_1) \cup E(\rho_2)$

and $G(\rho_1 \cup \rho_2) = G(\rho_1) \cup G(\rho_2)$ (take all arcs)

$\rho^+ = \bigcup_{i=1}^{n} \rho^i$ is the transitive closure of ρ (the smallest transitive relation containing ρ)

(*) If $\#(A) \leq n$ then $\rho^+ = \bigcup_{i=1}^{n-1} \rho^i$

If $\rho^n = \rho^{n-1}$ for some n, then $\rho^+ = \bigcup_{i=1}^{n-1} \rho^i$

(15) ρ^+ is also known as the reachability matrix, as it tells which vertices are reachable from any other vertex.

(17) EQUIVALENCE RELATIONS

ρ is an equivalence rel. on $A \Leftrightarrow \rho$ is reflexive, symmetric and transitive

Let ρ be an equiv. rel. on A. $a \in A$

Then $E_\rho(a) = \{x \mid x \rho a\}$ is called an equivalence class.

For simplicity, we write $[a] = E_\rho(a)$

Properties of $[a]$:

1) $[a] \cap [b] \neq \phi \Rightarrow [a] = [b]$
2) $[a] \neq \phi$ as $a \rho a$ (reflexive by defn)
3) $A = \bigcup_{a \in A} [a]$

(18) PARTITIONS OF A SET

Let $\{A_i\}_{i \in I}$ be a collection of subsets of A. Then $\{A_i\}$ forms a partition of A if:

1) $A_i \cap A_j \neq \emptyset \Rightarrow A_i = A_j$
2) $A_i \neq \emptyset$ for any i
3) $A = \bigcup_{i \in I} A_i$

A partition divides a set up into non-empty, non-intersecting subsets, whose union comprises the original set. Note that equivalence classes partition a set.

(19) QUOTIENT SETS AND QUOTIENT MAPS

Let $Q = \{[a] \mid a \in A\}$. Q is the quotient set of A under $\sim$, i.e. the collection of equivalence classes of A under $\sim$. The map $A \to Q$ which maps $a \to [a]$ is called the quotient map.

(24) (cf 33) FUNCTIONS

A function $f: A \to B$ from A to B is a mapping which is defined for all elements of A. A is the domain and B is the co-domain of f. A partial function is one in which the function is not defined for all elements of A. $f(a)$ denotes all the objects which f produces via a mapping. The range of f is $\{b \mid a \in A \Rightarrow b \in f\}$. Note that b must be unique for any a, else f is just a mapping, not a function. Let A, B be sets.

(26) Then: $A^B = \{f \mid f: B \to A\}$

$B^A = \{f \mid f: A \to B\}$

(27) INJECTIVE, BIJECTIVE AND SURJECTIVE FUNCTIONS

Let $f: A \to B$ be an arbitrary function

f is injective (one-to-one) if $f(a) = f(b) \Rightarrow a = b$.

f is surjective (onto) if $f(A) = B \Leftrightarrow$ for all $b \in B$, there is an $a \in A$ such that $b = f(a)$

i.e. $f(A) = B \Leftrightarrow$ onto $\Leftrightarrow \forall b \in B \; \exists a \in A \ni b = f(a)$

i.e. every element of B is mapped onto an element of A

(28) f is bijective if it is both one-to-one and onto

(29) IDENTITY FUNCTIONS

The identity function $I_A: A \to A$ is the function such that $I_A(a) = a$

COMPOSITE FUNCTIONS

Let $f: A \to B$, $g: B \to C$. Then define a new function $g \circ f: A \to C$ by setting for $a \in A$, $g \circ f(a) = g(f(a))$

(30) $g \circ f$ is the composite of f by g

INVERSE FUNCTIONS

Let $f: A \to B$ and $g: B \to A$.

Then g is the left inverse of f, i.e. $g \circ f = I_A$

and f is the right inverse of g, i.e. $g \circ f = I_B$

Inverse functions undo the effect of the function either on the left or right

$f: A \to B$

f has an inverse $\Leftrightarrow$ there is a unique function $g: B \to A$ such that $g \cdot f = I_A$ and $f \circ g = I_B$. We write $g = f^{-1}$

(33)
1) f is one-one, f has a left inverse (injective - left)
2) f is onto, f has a right inverse (surjective - right)
3) f is one-one and onto, f has a unique inverse.

(35) DATA TYPES (see also data structures, after logic)

A data type consists of a set and operations on that set. The set is called the set of values for the type.

(36) A literal is a constant whose value is given by its outward appearance

(37) A scalar type is a type which is not defined (but possibly definable) in terms of other types. A type which is not scalar (i.e. is defined in terms of other types) is a structured type. To define a type we must identify & describe the domain, and give a syntactic specification.

(38) SYNTACTIC SPECIFICATION OF OPERATIONS

In this case, we describe the form of operation. The number of variables on which an operation works is called the arity of the operation.

eg integer addition function '+' (a, b: integer): integer
↑ a 2-ary operation

A literal is a parameterless function, eg, maxint: → integer
a

(39) <u>SEMANTIC SPECIFICATION OF OPERATORS</u>

This tells what the operations do. Natural language is preferable provided it is precise enough. Usually give a general example.

(40) eg. LISP concatenate

concatenate two strings (ie puts them together)

$concatenate(x_1 x_2 \ldots x_m, y_1 y_2 \ldots y_n) = x_1 x_2 \ldots x_m y_1 y_2 \ldots y_n$

<u>ALGEBRAIC SPECIFICATION OF OPERATORS</u>

This is a form of semantic specification, where axioms are used to define the operator, and all needed properties can be deduced from these axioms. For more on algebraic spec, see textbook (Wulf, Shaw) and notes.

(44) <u>ABSTRACT MODELS.</u> (also semantic)

Here we define the domain and operations of one type in terms of other types (used by mathematicians). Thus all types described by abstract models are structured.

eg:

$$\mathbb{C} = \{ (a,b) \mid a,b \in \mathbb{R} \}$$

$$(a,b) + (c,d) = (a+c,\ b+d)$$

$$(a,b) \cdot (c,d) = (ac - bd,\ bc + ad)$$

Abs models = constructive defns = operational specs.

(45) # ORDERINGS

An ordering ρ on a set A is a relation such that

(a) $a \rho a$ (reflexive)

(b) $(a \rho b) \wedge (b \rho a) \Leftrightarrow (a = b)$ (ordered)

(c) $(a \rho b) \wedge (b \rho c) \Rightarrow (a \rho c)$ (transitive)

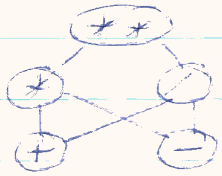
Usually we write $\leq$ rather than ρ.

A partial ordering occurs when two or more items are equivalent in terms of order but are not equal (eg operator precedence, * and / are same order but not equal, hence arithmetic operators are partially ordered). Orderings are often represented by Hasse diagrams, where vertices link adjacent elements in the ordering, and the largest element is at the top.

(ignore superfluous lines due to transitivity)

(46) Eg arithmetic operators $**$, $*$, $/$, $+$, $-$

Hasse diag:



LINEAR ORDERINGS AND WELL ORDERINGS

An ordering on A is linear or total if for $a, b \in A$, $a \neq b$ then either $a \leq b$ or $b \leq a$. Thus the integers are linearly ordered, ~~but the arithmetic operators are not~~

(47) A partial ordering on A is a well ordering if every subset of A has a smallest element. An element x in a set B is the smallest element if $x \leq \alpha \; \forall \; \alpha \in B$

Every linear ordering on a finite set is a well-ordering.

(48) Every well ordering is a linear ordering.

§4 NUMBERS

(49) COUNTABILITY AND DENUMERABILITY

A set is countable if its cardinality (i.e. $\#(\text{set})$) is $\aleph_0$ (aleph null) where $\aleph_0 = \#(\mathbb{N})$.

A set is denumerable if it is finite or countable.

A set A is infinite if there exists a set $B \subset A$ and a bijection $f: A \to B$ (bijection: one-one and onto)

That is, there exists a set B with the same cardinality which is contained within but not equal to A (e.g. $A = \mathbb{N}$, $B = \text{even } \mathbb{N}$)

If A and B are countable, then so is $A \times B$

If A " B are denumerable, then so is $A \times B$

Similarly, if $\{A_i\}_{i \in A}$ is a collection of countable (or denumerable) sets, then $\bigcup_{i \in \mathbb{N}} A_i$ is countable (or denumerable)

(50) Every non-finite subset A of a countable set B is countable

Let A, B be two sets. Then $\#(A) \leq \#(B)$ if there is a one-one map $f: A \to B$

§5 THE NATURAL NUMBERS

(51) INDUCTION

THE NATURAL NUMBERS AND PEANO'S AXIOMS

The natural numbers are formed from a set denoted $\mathbb{N}$, a constant $1 \in \mathbb{N}$, and a map $S : \mathbb{N} \to \mathbb{N}$ called the successor function, such that:

(a) S is one-one

(b) $\neg(\exists n \ni S(n) = 1)$ i.e., there is no number $\in \mathbb{N}$ whose successor is 1.

(c) Let $T \subseteq \mathbb{N}$. Then $(1 \in T) \wedge (n \in T \Rightarrow S(n) \in T)$ $\Rightarrow T = \mathbb{N}$

This is the PRINCIPLE OF INDUCTION.

These are called PEANO'S AXIOMS.

From (c) follows the principle of recursive definition. Suppose one wishes to define some object or property $P(n)$ for each $n \in \mathbb{N}$. Then one need only define $P(1)$ and then define $P(n)$ based on the definition of $P(n-1)$. By (c) $P(n)$ would be defined for all $n \in \mathbb{N}$

(53) PROPERTIES OF THE NATURAL NUMBERS

INTEGER DIVISION: Let $m, n \in \mathbb{N}$, $m \neq 0$.

Then $n = qm + r$ where $0 \leq r < m$

q is called the quotient, r the remainder.

We say m divides n (n is a multiple of m) if $r = 0$

We say n and m are relatively prime if $gcd(n,m) = 1$

THE EUCLIDEAN ALGORITHM FOR $\mathbb{N}$

Let $m, n \in \mathbb{N}$; $m \geq n$

Then $m = q_1 n + r_1$

If $r_1 = 0$, then $g.c.d.(m,n) = n$

Else $n = q_2 r_1 + r_2$

If $r_2 = 0$ then $g.c.d.(m,n) = r_1 = g.c.d.(n, r_1)$

Else $r_1 = q_3 r_2 + r_3$ etc.

Then $g.c.d.(m,n)$ = last non-zero remainder.

If $\frac{mn}{p}$ and p is prime, then $\frac{m}{p}$ or $\frac{n}{p}$

We can express $g.c.d.(m,n)$ in the form:

$g.c.d.(m,n) = jn + km$ for some $j, k \in \mathbb{Z}$

by proceeding backwards. Say that

$r_3 = g.c.d.(m,n)$ in last example

Then $r_3 = r_1 - q_3 r_2$

$= r_1 - q_3(n - q_2 r_1)$

$= (m - q_1 n) - q_3(n - q_2(m - q_1 n))$

$= m - q_1 n - q_3 n + q_2 q_3 m - q_1 q_2 q_3 n$

$= (1 + q_2 q_3)m - (q_1 + q_3 + q_1 q_2 q_3)n$ $\rightarrow$

(55) ## PRIME FACTORISATION THEOREM

Any $n \in \mathbb{N}$ can be represented as a unique product of

powers of primes. Eg: $72 = 2^3 \cdot 3^2$

$108 = 2^2 3^3$

Note that $l.c.m(m,n) = \frac{nm}{g.c.d(m,n)}$

Given $m = p_1^{m_1} p_2^{m_2} p_3^{m_3} \ldots$

$n = p_1^{n_1} p_2^{n_2} p_3^{n_3} \ldots$

We can show: $g.c.d(m,n) = p_1^{\min(m_1,n_1)} p_2^{\min(m_2,n_2)}$ etc

$l.c.m(m,n) = p_1^{\max(m_1,n_1)} p_2^{\max(m_2,n_2)}$ etc

As an example: $g.c.d.(72,108) = 2^2 3^2$

$l.c.m.(72,108) = 2^3 3^3$

We can use this to simplify rational numbers expressed as quotient. If $g.c.d.(m,n) < m$ and $g.c.d(m,n) < n$, then $\frac{n}{m}$ can be simplified to $\frac{(n/g.c.d)}{(m/g.c.d)}$.

BASE REPRESENTATION

Let $b \geq 2$ (b = base) and $n \in \mathbb{N}$.

Then n can be expressed uniquely as

$$n = a_1 + a_2 b + a_3 b^2 + a_4 b^3 + \ldots + a_m b^{m-1}$$

where $0 \leq a_i < b$

a_1 is the remainder on dividing n by b

a_2 is the " " " first quotient by b, etc

§7 BOOLEAN ALGEBRA

We define a small set of axioms sufficient to deduce the theorems of this type of system:

DEFN 1: A *Boolean algebra* is a triple $(S, \vee, \wedge)$ where S is a set and $\vee$ and $\wedge$ are two functions $\vee, \wedge : S \times S \to S$ satisfying:

Boolean ①	Commutative	$x \vee y = y \vee x$ ‖ $x \wedge y = y \wedge x$
②	Distributive	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ ‖ $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
Group ③	Identity	$\exists 0 \in S \mid 0 \vee x = x \vee 0 = x$ ‖ $\exists 1 \in S \mid x \wedge 1 = 1 \wedge x = x$
④	Complement	$\forall x \exists x' \mid x' \vee x = 1 = x \vee x'$ ‖ $\forall x \exists x' \mid x \wedge x' = x' \wedge x = 0$

From this, we can note the *Principle of Duality*: for every law, we can exchange $\vee$ and $\wedge$, and 0 and 1 and preserve identity (constants become their pairs, variables remain the same)

THM 1: (a) 0 and 1 are unique elements satisfying identity ③
(b) given x, x' is unique

Proof by r.a.a.

THM 2: ⑤ $x'' = x$ (involution) ‖ $x'' = x$
⑥ $x \vee x = x$ (idempotent) ‖ $x \wedge x = x$
⑦ $x \vee 1 = 1$ (null) ‖ $x \wedge 0 = 0$
⑧ $x \vee (x \wedge y) = x$ (absorption) ‖ $x \wedge (x \vee y) = x$

THM 3: ⑨ $x \vee (y \vee z) = (x \vee y) \vee z$ (associativity) ‖ $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

THM 4 (De Morgan): ⑩ $(x \vee y)' = x' \wedge y'$ ‖ $(x \wedge y)' = x' \vee y'$

COR $0' = 1$, $1' = 0$

DEFN x *precedes* y $(x \leq y)$ *iff* $(x \vee y) = y$

eg for sets $A \leq B$ iff $A \cup B = B$ similar.

- $\leq$ is a Partial Ordering (P.o.) relation
- $x \vee y = \text{lub}\{x, y\}$ $x \wedge y = \text{glb}\{x, y\}$

DEFN Any P.O set where $\exists$ a glb and lub for any two elements is called a *lattice*

- A lattice satisfying ②, ③ & ④ is a Boolean algebra
- conversely, a B. alg. defines a lattice by ① & ②.

(a Bool alg is a lattice defined as $a \leq b \Leftrightarrow a \wedge b = a$)

DEFN A *boolean expression* in variables $x_1 \ldots x_n$ over a Bool. alg $(S, \wedge, \vee)$ is defined by:

- all elements of S as well as $x_1, \ldots x_n$ are bool expns.
- if e_1 and e_2 are expressions, so are:

$$(e_1 \vee e_2)$$
$$(e_1 \wedge e_2)$$
$$(e_1')$$

(Boolean expressions can also be defined recursively adding constants $\alpha_1 \ldots \alpha_n$ which themselves are B. expns.

IDENTITY

Identity is a relation – two Bool. expns. are identical if they can be made equivalent by the principle of duality.

DEFN B^* is a sub Bool. alg. if $B^* \subseteq B$

$$x, y \in B^* \Rightarrow x \vee y,\ x \wedge y \in B^*$$

$$A \subseteq B \Rightarrow (2^A, \vee, \wedge) \text{ is a sub B. alg of } (2^B, \vee, \wedge)$$

$f: (B, \vee, \wedge) \to (\bar{B}, \bar{\vee}, \bar{\wedge})$ is a map of B. alg. if

(a) $f: B \to B'$

(b) $f(a \vee b) = f(a) \,\bar{\vee}\, f(b)$

(c) $f(a \wedge b) = f(a) \,\bar{\wedge}\, f(b)$

(d) $f(0) = \bar{0}$, $f(1) = \bar{1}$

Let $(P, \leq)$ be a P.O. set. Let $A \subset P$.

A has a minimum element m if $(m \in A) \wedge \forall x \in A,\ m \leq x$

Recall a Bool alg is lattice defined as $a \leq b \Leftrightarrow a \wedge b = a$, and a lattice is defined by a set having a glb and lub for any two elements. Thus, in a B. alg, we write:

$$l.u.b\ A = \bigvee_{a \in A} a \quad \text{(minimum)}$$

$$l.u.b\ \{a, b\} = a \vee b$$

$$g.l.b.\ A = \bigwedge_{a \in A} a \quad \text{(maximum)}$$

$$g.l.b\ \{a, b\} = a \wedge b$$

Let P be a lattice where $\bigvee_{i \in I} a_i$ for all collections $\{a_i\}_{i \in I}$

and where $b \wedge \left(\bigvee_{i \in I} a_i\right) = \bigvee_{i \in I} (a \wedge a_i)$

then P is called a <u>locale</u> (generalization of topological space)

6) THE RATIONAL AND THE REALS

We defined $\mathbb{N}$ in terms of sets and Peano's axioms, and from this $\mathbb{N}^-$ set of negative integers.
We can use $\mathbb{Z}$ to define the rationals $\mathbb{Q}$ and the reals $\mathbb{R}$.

DEFN 1 Rational numbers are $S = \mathbb{Z} \times (\mathbb{Z} - \{0\})$ (we think of $p, q \in S$ as p/q)

on $S \times S$, $(m,n) \equiv (p\,q)$ if result after cancelling all common factors in (m,n) = cancelled (p,q), where $\equiv$ is an equivalence relation

The rationals $\mathbb{Q}$ are defined to be $S/\equiv$. So a rational number $\frac{p}{q}$ can be considered to be a class of pairs under equivalence.
We write $\frac{p}{q} = (p,q)$ and the equivalence class of p/q as $\left[\frac{p}{q}\right]$

We define $\left[\frac{p}{q}\right] \pm \left[\frac{m}{n}\right] = \left[\frac{pn \pm qm}{qn}\right]$

$$\left[\frac{p}{q}\right] \cdot \left[\frac{m}{n}\right] = \left[\frac{pm}{qn}\right]$$

With these definitions, the rationals form a field, the zero being $\left[\frac{0}{1}\right]$ and the identity being $\left[\frac{1}{1}\right]$

Order is defined by $0 \leq \left[\frac{p}{q}\right]$ if p and q have the same sign or $p = 0$

$$\left[\frac{p}{q}\right] \leq \left[\frac{m}{n}\right] \text{ iff } 0 \leq \left[\frac{m}{n}\right] - \left[\frac{p}{q}\right]$$

This satisfies the axioms of a total ordering

Thus the rationals form an ordered field, and can be defined as equivalence classes of pairs of integers.

DEFN 2 The real numbers are defined as follows:

A set S is called a real number if:

(i) $S \subset \mathbb{Q}$

(ii) S is bounded below

(iii) if $q \in S$ then for all $q' \in \mathbb{Q}$ s.t. $q \leq q'$ we have $q' \in S$

So if S is an open infinite segment bounded below (Dedekind section)
if glb of S exists and is s (say), then we call S
rational and treat it as s
but if glb S does not exist, then we call S irrational.

Addition, multiplication and order can be defined, and the resulting set is a complete ordered field.

LOGIC

If p and q are propositions, we have the following logical connectives:

$p \vee q$	disjunction
$q \wedge q$	conjunction
$\neg p$	negation
$p \Rightarrow q$	implication
$p \Leftrightarrow q$	equivalence (mutual implication)

Let P be a class of propositions satisfying
- if $p, q \in P$ then $p \vee q,\ p \wedge q,\ \neg p,\ \neg q \in P$

To ensure the map $v : P \to \underline{2}$ is well defined, we must impose an algebraic structure on $\underline{2}$:

$v : P \to \underline{2}$ is a <u>valuation</u> or <u>evaluation</u> satisfying:

$$v(p \vee q) = v(p) \vee v(q)$$
$$v(p \wedge q) = v(p) \wedge v(q)$$
$$v(\neg p) = \neg v(p)$$

if $v(p) = 1$, then p is valid for v, or $v \vdash p$

A <u>tautology</u> is a proposition which is valid for any valuation.

THE LINDENBAUM ALGEBRA

A B.alg $(B, \vee, \wedge, \neg, 0, 1)$ is an algebraic structure with two binary operations $\vee$ and $\wedge$ and one unary operation $\neg$. 0 and 1 are the special elements of B. Every B.alg satisfies the following identities:

① $x \vee (y \vee z) = (x \vee y) \vee z$ (Associativity) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

② $x \vee y = y \vee x$ (Commutativity) $x \wedge y = y \wedge x$

③ $x \vee x = x$ (Idempotent) $x \wedge x = x$

④ $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ (Distributive) $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

⑤ $x \vee (x \wedge y) = x$ (Absorption) $x \wedge (x \vee y) = x$

⑥ $\neg(x \vee y) = (\neg x) \wedge (\neg y)$ (de Morgan) $\neg(x \wedge y) = (\neg x) \vee (\neg y)$

⑦ Identities: $x \vee 0 = x$ $x \wedge 0 = 0$ $x \wedge (\neg x) = 0$

$x \vee 1 = 1$ $x \wedge 1 = x$ $x \vee (\neg x) = 1$ $0 \neq 1$

⑧ Double negation: $\neg\neg x = x$

Let A and B be B algs. A map $f: A \to B$ is called a homomorphism if

$f(x \vee y) = f(x) \vee f(y)$

$f(x \wedge y) = f(x) \wedge f(y)$

$f(\neg x) = \neg f(x)$

$f(0) = 0$

$f(1) = 1$

Let A and B be B algs. A binary relation $\leq$ is an order relation on a B alg if $x \leq y \Leftrightarrow x \vee y = y$ and $x \wedge y = x$. Any homomorphism $f: A \to B$ preserves the order, that is, $x \leq y \Leftrightarrow f(x) \leq f(y)$. Note the connection between a homomorphism and a valuation of a Boolean algebra.

Similarly we can define the equivalence relation $\equiv$ as $p \equiv q \Leftrightarrow (v(p) = 1)$ and $(v(q) = 1)$ are equivalent

TAUTOLOGIES AND VALID ARGUMENTS

P is a tautology if it is valid for any valuation, ie, $\neg p$ is logically false. An argument is a finite series of propositions $p_1 \ldots p_n\ q$, where $p_1 \ldots p_n$ are premises and q is the conclusion. The argument is valid (q is deducible/provable from $p_1 \ldots p_n$) iff $(p_1 \wedge p_2 \ldots \wedge p_n) \rightarrow q$ is a tautology. To show this, we can use the rule of inference (Modus Ponens): $p \wedge (p \rightarrow q) \rightarrow q$.

To test validity we can either use truth tables or the properties of B. algs, or the following method:

- using assumptions (fantasies) and tautologies
- modus ponens
- contradiction
- transitivity $(p \rightarrow q,\ q \rightarrow r,\ \Rightarrow p \rightarrow r)$

QUANTIFIERS

Expressions containing variables or quantifiers are called propositional forms, as they are capable of generating propositions

THM: Let $S(x)$ be any propositional form. Then:
$\forall x\ [S(x)] \rightarrow S(b)$ is a tautology (b any name)

DEFNS: Let $S(x)$ be any propositional form.
Then $\forall x\ [S(x)]$ is true iff $S(x)$ is true for any x.
and $\exists x\ [S(x)]$ is true iff $S(x)$ is true for at least one x.

Note that $L = \langle P/\equiv, \vee, \wedge, \neg, \forall, \exists \rangle$ is a Tarskian algebra after adding the quantifiers and defining $p \equiv q \Leftrightarrow \vdash p \Leftrightarrow q$ (i.e. $p \to q$ and $q \to p$ is a tautology)

THm

① $v(\neg \forall x (S(x))) = 1$ iff $v(\exists x (\neg S(x))) = 1$

② $v(\neg \forall x (\neg S(x))) = 1$ iff $v(\exists x (Sx)) = 1$

③ $v(\neg \exists x (\neg S(x))) = 1$ iff $v(\forall x (S(x))) = 1$

④ $v(\neg \exists x (S(x))) = 1$ iff $v(\forall x (\neg S(x))) = 1$

THm Let $S(x)$ and $R(x)$ be propositional forms. Then

$\exists x (S(x) \wedge R(x)) \to \exists x (S(x)) \wedge \exists x (R(x))$ is a tautology

PROPOSITIONS INVOLVING SEVERAL QUANTIFIERS

If S is an n-place holder propositional form, and $Q_1 \dots Q_n \in \{\exists, \forall\}$, then $Q_1(x_1) \dots Q_n(x_n)(S(x_1, \dots x_n))$ is a proposition

THm Let $S(x, y)$ be any propositional form involving two place-holders. Then the proposition:

$\exists x \forall y (S(x, y)) \to \forall y \exists x (S(x, y))$

is a tautology

DATA STRUCTURES

DEFNS: A value is an element of some set associated with a type

A variable is a component of memory and has both name and type associated with it

A memory consists of a collection of variables and 2 operations, fetch and store

A type consists of a set of values and operations

Memory can be described functionally or axiomatically:

FUNCTIONAL DESCRIPTION OF MEMORY

A mapping of names of individual memory cells to the values in them is used:

memory : name → value

memory(name) → value

domain = {name} range = {value}

Thus memory can be considered to be a set of ordered pairs, with a mapping function linking each pair of elements.

On assignment operation produces a new mapping function, so assignment is 'functional'.

assignment : (name → value) × name × value → (name → value)

(name → value): function; name × value: arguments; (name → value): function

AXIOMATIC DESCRIPTION OF MEMORY

As it is possible for variables to have aliases (eg in arrays) we must account for all possible aliases for the variable being changed.

We must introduce a way to identify the new state of the 'array' after assignment

DEFN If A is an array $[Q]$ of type P, and i is a value of type Q, e a value of type P, then: $\alpha(A, i, e)$ is the value of A after the assignment $A[i] = e$. From this follows

THM $$\alpha(A, i, e)[j] = \begin{cases} e & \text{if } i = j \\ A[j] & \text{if } i \neq j \end{cases}$$

We can now define an assignment axiom:

$$wp(A[i] := E, Q[A]) \equiv (E \text{ is defined and } Q(\alpha(A, i, E)))$$

(where $wp(S, Q)$ is the weakest precondition, i.e. necessary & sufficient condition, that assures the proper termination of statement S in a state satisfying predicate Q)

The two descriptions of memory are identical both producing a new <name, value> set from an old one.

ALGEBRAIC SYSTEMS.

We will give the main features of most of the algebraic systems we have encountered, and obtain some new results.

DEFN: A semigroup $S = \langle S, \circ \rangle$ is a set S and a function $\circ : S \times S \to S$ (called the product) satisfying $(x \circ y) \circ z = x \circ (y \circ z) \left(\forall \begin{smallmatrix} x \\ y \end{smallmatrix} \in S\right)$ (associative)

IDENTITY: A monoid is a semigroup $\langle S, \circ \rangle$ with an element $e \in S$ s.t. $x \circ e = e \circ x = x$ for all $x \in S$

For example, $S = X^+ = \{$nonempty strings of elements of $X\}$ where $\circ$ is concatenation has a semigroup. $\langle S, \circ \rangle$ If $S' = X^* = \{$strings of elements of $X\} = X^+ \cup \{\Lambda\}$, then $\langle S', \circ \rangle$ is a monoid. (Λ is the identity element, a null string) It is called a free monoid as $W_1 \circ W_2 = \Lambda \Rightarrow W_1 = W_2 = \Lambda$

Note: For semigroups, $x^p = x \circ \ldots \circ x$ is well defined because of the associative law, and $x^p \circ x^q = x^{p+q}$ (where p, q are +ve integers > 0)

THM 1: A monoid has a unique identity

DEFN: A group is a monoid $\langle S, \circ \rangle$ which satisfies inverses $(\forall x \in S)(\exists x^{-1} \in S) \ni (x \circ x^{-1} = x^{-1} \circ x = e)$

i.e., a group is a set with a binary operation satisfying the associative law, (9 laws), having an identity element and for which each element has an inverse

DEFN A group is finite if it has finitely many elements

An abelian "object" has operation $\circ$ satisfying
$g \circ f = f \circ g \quad \forall g, f \in S$ ($\circ$, commutative)

THM 2 (a) The inverse element of any x in the group is unique
(b) $a \circ x = b$ and $x \circ a = b$ have unique solutions x in a group
(c) $a \circ b = a \circ c \Rightarrow b = c$ (cancellation)

PROOF: (a) Suppose $x \circ a = e$ and $b \circ x = e$
Then $b \circ (x \circ a) = b \circ e = b$
$\Rightarrow (b \circ x) \circ a = e$
$\Rightarrow e \circ a = b$
$\Rightarrow a = b$

(b) $a \circ x = b \Rightarrow a^{-1} \circ a \circ x = a^{-1} \circ b \Rightarrow x = a^{-1} \circ b$ (more unique)
$a \circ (a^{-1} \circ b) = b \Rightarrow x = a^{-1} \circ b$ is a solution (prove existence)
similarly for $x \circ a = b$

(c) $a \circ b = a \circ c \Rightarrow a^{-1} \circ a \circ b = a^{-1} \circ a \circ c$
$\Rightarrow e \circ b = e \circ c \Rightarrow b = c$

PROPERTIES OF GROUPS

DEFN: A subgroup $\tilde{G}$ of a group $G = \langle S, \circ \rangle$ is a subset $\tilde{S}$ of S which is such that $\langle \tilde{S}, \circ \rangle$ is a group. The subgroup is proper if $\tilde{G} \neq \{e\}$ and $\tilde{G} \neq G$

THM 3 Let $G = \langle S, \circ \rangle$ and let $\tilde{S} \in S$. Then $\langle \tilde{S}, \circ \rangle$ is a subgroup iff:

(a) $x \circ y \in \tilde{S}$ for all $x, y \in \tilde{S}$ (closure)

(b) $\forall x \in \tilde{S}$, $x^{-1} \in \tilde{S}$. (inverses)

Note - identity element need not be in the subgroup.

DEFN Let $\tilde{G}$ be a subgroup of G. For $g \in G$, let $g\tilde{G} = \{gx \mid x \in \tilde{G}\}$. Then $g\tilde{G}$ is called a left coset of g w.r.t. $\tilde{G}$

THM 4 Let $\tilde{G}$ be a subgroup of G. Then $\{g\tilde{G} \mid g \in G\}$ is a partition of G.

Proof Since $\tilde{G}$ is a subgroup, $e \in \tilde{G}$ and hence for each $g \in G$, $g = ge \in \bigcup_{g \in G} g\tilde{G}$

Thus $G \subseteq \bigcup_{g \in G} g\tilde{G}$, and since the reverse inclusion is trivial, $G = \bigcup_{g \in G} g\tilde{G}$

Clearly $g\tilde{G}$ is not empty for any g, since $\tilde{G}$ must have at least one member (namely e).

Suppose $g_1\tilde{G} \cap g_2\tilde{G} \neq \emptyset$, and let $g \in g_1\tilde{G} \cap g_2\tilde{G}$
Then there exists $x, y \in \tilde{G}$ $\ni$ $g = g_1 x$ and $g = g_2 y$
thus $g_1 = g_2 y x^{-1}$

Let $z \in \tilde{G}$ and consider $g_1 z$:

$$g_1 z = (g_2 y x^{-1}) z = g_2 (y x^{-1} z)$$

Since $x, y, z \in \tilde{G}$ and $\tilde{G}$ is a subgroup, $yx^{-1}z \in \tilde{G}$
so $g_1 z \in g_2 \tilde{G}$.
Similarly, for arbitrary $z \in \tilde{G}$, $g_2 z \in g_1 \tilde{G}$. Hence $g_1\tilde{G} = g_2\tilde{G}$
Thus $g_1\tilde{G}$ and $g_2\tilde{G}$ are either identical or disjoint, thus the set is partitioned

Note The equivalence relation corresponding to the partition is $g_1 \underset{\tilde{G}}{\equiv} g_2$ iff $\exists x, y \in \tilde{G}$ such that $g_1 x = g_2 y$.

PROPOSITION If $\#(\tilde{G})$ is finite, $\#(g\tilde{G}) = \#(\tilde{G})$

Proof Let $\tilde{G} = \{g_1, \dots, g_n\}$. Then $g\tilde{G} = \{gg_1, \dots, gg_n\}$. Thus $\#(\tilde{G}) \geq \#(g\tilde{G})$. However, $>$ can only hold if $gg_i = gg_j$ for some i, j with $i \neq j$. But by multiplying each by g^{-1}, we get $g_i = g_j$, hence $>$ cannot hold. Thus $\#g\tilde{G} = \#\tilde{G}$.

THEOREM 5 (Lagrange)

Let G be a finite group. Let there be n distinct cosets $g_1\tilde{G}, \dots, g_n\tilde{G}$ of the subgroup $\tilde{G}$ of G. Then $\#G = n(\#\tilde{G})$

Proof $G = g_1\tilde{G} \cup \dots \cup g_n\tilde{G}$ by partition. Since these terms are distinct

$$\#G = \#g_1\tilde{G} + \dots + \#g_n\tilde{G}$$

But $\#g\tilde{G} = \#\tilde{G}$ by proposition

$$\Rightarrow \#G = n\,\#\tilde{G}$$

COROLLARY Let G be a finite group:

(a) The number of elements of a subgroup is a divisor of the number of elements in the group

(b) The number of distinct cosets with a subgroup is a divisor of the number of elements in the group.

COROLLARY Any group of prime order has no proper subgroups

DEFN The number of elements in a group is called its order

THM 6 Let G be a finite group. Then $\forall g \in G \; \exists r \to g^r = e$ where r is a positive integer

Proof Since G is finite, $\{g, g^2, g^3, \ldots\}$ cannot have infinitely many distinct elements, hence $g^m = g^n$ for some $m, n \geq 0$ and $m \neq n$. Then $g^m g^{-n} = g^n g^{-m} = e$ i.e. $g^{m-n} = g^{n-m} = e$. Either $m-n$ or $n-m$ is positive, thus $r = |m-n|$.

DEFN The smallest integer $r \geq 1$ such that $g^r = e$ is called the order of the element g.

THM 7 Let G be finite, $g \in G$. Then if r is the order of g, $\langle S, \circ \rangle = \langle \{e, g, g^2, \ldots, g^{r-1}\}, \circ \rangle$ is a subgroup of order r. This is called the cyclic (sub)group generated by g

COR The order of an element of a finite group divides the order of the group

Proof The order of g is r, which is also the order of the cyclic subgroup, and as the order of a subgroup divides the order of the group, r divides the order of the group.

COR Let $p \in \mathbb{N}$ be prime, $a \not\equiv 0 \pmod p$. Then $a^{p-1} \equiv 1 \pmod p$

Proof $S = \{1, \ldots, p-1\}$ forms a group under multiplication (mod p). Let $a \equiv b \pmod p$ where $b \in S$

The order of the group is $p-1$, hence if r is the order of b ($i.e.\ b^r = 1$), we have $p-1 = kr$ for some k.

Thus $b^{p-1} = b^{kr} = (b^r)^k \equiv 1^k = 1 \pmod p$

~~and~~ $b^{p-1} \equiv 1 \pmod p$ and thus $a^{p-1} \equiv 1 \pmod p$

FIELDS

A field is a tuple $\langle S, \cdot, + \rangle$ where $\cdot, +$ are binary operations such that $\langle S, + \rangle$ is an Abelian group and $\langle S - \{0\}, \cdot \rangle$ is an Abelian group, and the distributive law holds

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad (a, b, c \in S)$$

VECTOR SPACES

We have dealt with these at length in other sections. Note that it is not necessary to use $\mathbb{R}$ or $\mathbb{C}$ as our scalar multipliers - any field could have been used.

MORPHISMS

Two groups G_1 and G_2 are isomorphic if there exists a map $f : G_1 \to G_2$ satisfying:

(a) f is a bijection

(b) $f(g \circ h) = f(g) \circ f(h)$

($g \circ h$: product in G_1; $f(g) \circ f(h)$: product in G_2)

f is called an isomorphism.

THM 8 (a) If f is an isomorphism from G_1 to G_2, then f^{-1} is an isomorphism from G_2 to G_1.

(b) If G_1, G_2 are isomorphic by f, and e is the identity in G_1, e' in G_2, then $f(e) = e'$

(c) If f is an isomorphism, $f(x^{-1}) = f(x)^{-1}$

GROUP TABLES

A group is defined by its multiplication tables which can be enumerated if group is finite and small. We must check associativity, inverse and identity to be sure we have a group. If the table is symmetric about the main diagonal it is Abelian. We can also use group tables to check whether groups are isomorphic iff by simply changing the symbols representing elements of one group we can transform its group table to that of the other

SYMMETRY

This is one of the chief applications of group theory. We will describe the symmetry of plane polygons. A plane polygon can be regarded as a bounded set of points in the plane whose boundary is made up of straight line segments. We will think of the vertices as numbered in some order and we will consider transformations of the polygon obtained by moving the whole figure without distortion so that some or all of

the vertices have changed position. If the set of points obtained by such a transformation is the same as the original set, the transformation is called a ~~symmetry~~ of the polygon.

Eg Equilateral $\triangle$ 

The only symmetry (besides the identity $\begin{pmatrix}1&2&3\\1&2&3\end{pmatrix}$), is $\begin{pmatrix}1&2&3\\1&3&2\end{pmatrix}$,

For equilateral triangles all permutations are symmetries. It is easily seen that the symmetries of a polygon form a group, as all of the properties of groups hold.

Note The symmetry group of a regular polygon with n vertices is called the dihedral group of order $2n$.

We now consider the 13 new types of groups.

① $\{0,1\}^+$ • concatenation
Not monoid as no identity element exists, hence semigroup

② $\{0,1\}^*$ • concatenation
Monoid but no inverse thus not a group.

③ $\mathbb{N}$ + addition
$0 \notin \mathbb{N}$ thus a semigroup, not monoid

④ $\mathbb{N} \cup \{0\}$ + addition
Monoid, but not a group (no additive inverse)

⑤ $\mathbb{Z}$ + addition
Abelian group.

(8) $M_2(R)$ (2×2 matrices) under matrix addition
Abelian group $I = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ Inverse $= \begin{pmatrix} -a & -b \\ -c & -d \end{pmatrix}$

(9) $M_2(R)$ under matrix multiplication
Monoid, associative $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Not a group, as not all M_2 have multiplicative inverses

(10) $GL_2(R)$ (invertable 2×2 mat.) under mat mult.
General linear group, but not Abelian.

(11) $SL_2(R)$ (2×2 having det ±1) under mat mult.
Special linear group, invertable because of non zero det.

(12) $\mathbb{Z}$ under multiplication
associative, 1 identity but no inverses, thus a monoid.

(13) $GL_2(R)$ under +
Not a group, as $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ is not an element yet is the result of adding inverses.

(14) $\{\mathbb{Q} - \{0\}, \cdot\}$
Abelian, ~~add~~ $(\mathbb{Q}, +, \cdot)$ $(R, +, \cdot)$ $(\mathbb{C}, +, \cdot)$ fields.

(15) $End(\mathcal{E}) = \{f : \mathcal{E} \to \mathcal{E}\}$ where $\mathcal{E}$ is a set
ie, the collection of maps from $\mathcal{E}$ to $\mathcal{E}$.
with operation of composition $f \circ g(x) = f(g(x))$
This is associative, with an identity (the identity map) $End(\mathcal{E})$ is called the collection of endomorphisms of $\mathcal{E}$
Provided f is bijective (one-one & onto) so has inverses then we get a group, called $Aut(\mathcal{E})$, or the automorphisms of $\mathcal{E}$.

GENERAL

Symmetries can be described as matrices (initial / mapped) or enumerated as sets. In general, order of a group of n distinct elements is $n!$

FINITE STATE MACHINES.

DEFN A FSM is a 4-tuple $(Q, q_0, NEXT, OUT)$,
where: Q is a finite set of possible states
$q_0 \in Q$ is the initial state
NEXT is a state-transition function mapping
<state, input symbol> pairs to states
OUT is a function that maps <state, input symbol>
pairs to output symbols.

STATE TRANSITION DIAGRAMS.

Each node (circle) represents a state, with an arc leading out of it for each legal input in that state. Each arc points to the node representing the NEXT map, and is labeled with a phrase of the form $x \rightarrow y$ where x is the input value and y the corresponding output. The initial state is usually represented by an arrow that comes from nowhere.

FINITE STATE RECOGNISERS.

An FSR is a 4-tuple $(Q, q_0, NEXT, F)$ where $F \subset Q$ is a set of final states. If after processing its input the FSR is in a final state, it has recognised its input, else it has rejected it. In STD's, the final states are indicated by double circles.

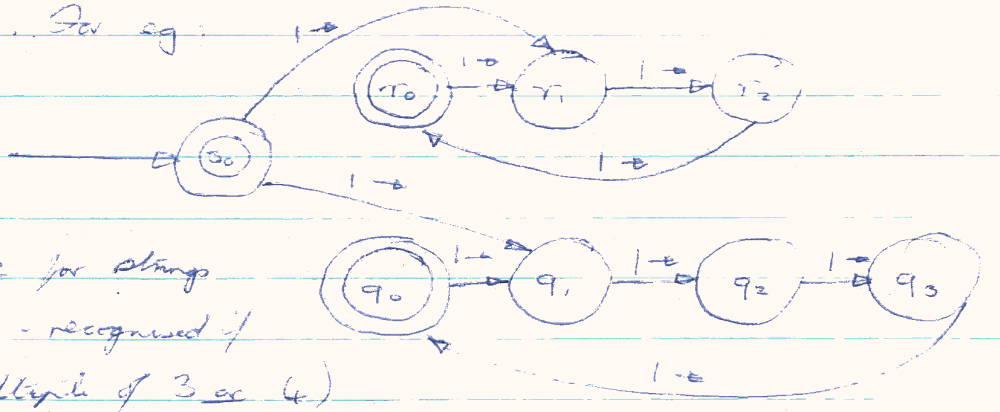
FINITE STATE GENERATORS

These accept no input (i.e. null string ε) and simply have output arcs $(\rightarrow y)$

By allowing ε as a possible result of OUT and possible arguments of OUT and NEXT, we can create FSM's which are mixtures of FSR's & FSG's. The input and output alphabets have no restrictions and can be defined as anything.

NONDETERMINISTIC FSM's

A NDFSM accepts an input string if any of the possible paths through the states indicated by their arcs and their labels leaves the machine in a final state. For eg:



(NDFSR for strings of 1's - recognised if a multiple of 3 or 4)

An NDFSR, N, is a 4-tuple (Q, q_0, NEXT, F)

Where: Q is a finite set of states

$q_0 \in Q$ is the initial state

$F \subset Q$ is the set of final states

NEXT is a function defined on certain pairs (q, a) of states and inputs (a may be ε) and

yields sets of possible next states (subsets of Q)
If NEXT is defined for the pair (q, ε) then it is an NDFSR, i.e., NEXT (q, b) is undefined where b is any non-null input.

A computation of a NDFSR is a sequence of states $S_0, S_1, \ldots S_n$ where
- $S_0 = q_0$
- S_{i+1} is in NEXT (S_i, b) where b is the current input
- Input is consumed at or before S_n (possibly before if there are null transitions)

An NDFSR accepts an input X iff some computation ends in an accepting (final) state.

EQUIVALENCE OF NDFSR'S AND FSR'S

For any NDFSR, there exists an FSR that accepts exactly the same set of inputs. To prove this, we can construct an FSR (however, this is not necessarily the least equivalent FSR).

Let $N = (Q, q_0, F, \text{NEXT})$ be an NDFSR. We assume without loss of generality that N has no null transitions. We construct our FSR N' where $N' = (Q', q_0', F', \text{NEXT}')$ as follows:

$$Q' = 2^Q$$

$$q_0' = \{q_0\}$$

$$\text{NEXT}'(\{q_1, \ldots, q_r\}, a) = \text{NEXT}(q_1, a) \cup \text{NEXT}(q_2, a) \cup \ldots \cup \text{NEXT}(q_r, a)$$

$$F' = \{q' \in Q' \mid q' \cap F \neq \phi\}$$ (i.e. all subsets containing at least one final state)

LANGUAGES

A language is a set of strings of symbols. Each string from the language is called a sentence (or word) of the language. The set of symbols of the language is called its alphabet or vocabulary - we refer to a language over an alphabet.

REGULAR EXPRESSIONS

DEFN Let $A = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. Then a R E over A denotes a language over A and is defined by the following set of rules:

- atom: any single symbol of A (including ε) is a R.E.
- alternation: If R_1 and R_2 are both R E's, then $(R_1 + R_2)$ is a valid R.E (Eg $(a_1 + a_2) = \{a_1, a_2\}$)
- composition: If R_1 and R_2 are both R.E's, then so is $R_1 R_2$ (Eg $(a_1 a_2) = \{a_1 a_2\}$)
- closure: If R_1 is a R.E, then so is $(R_1)^*$ (eg $(a_1)^* = \{\varepsilon, a_1, a_1 a_1, \ldots\}$)

(Note, the notation $(a)^+ \equiv a(a)^*$, ie, is the same as $(a)^*$ except for the ε.)

- nothing else is an R.E.

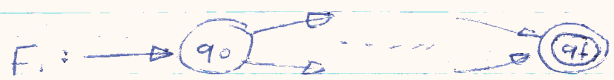
The hierarchical rules for these are closure first, followed by composition and then alternation.

FSM AND RE EQUIVALENCE.

ITEM: Any language that can be denoted by a R.E. can be recognised by a FSR

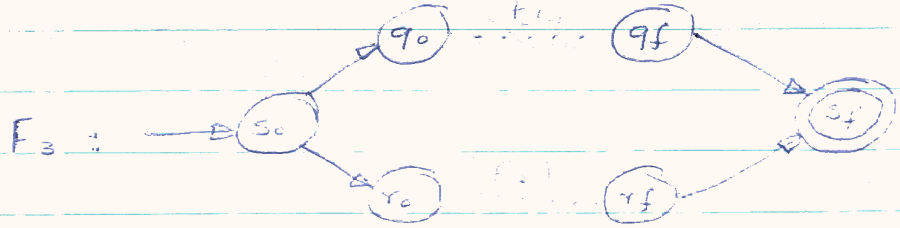
In fact, the class of languages denoted by R.E's ≡ the class of languages recognised by FSR's. Again we can prove by construction

To do this, assume that we will be constructing our NDFSR's out of other NDFSR's that have been previously constructed (recursive). Call these F_1 and F_2, with initial states q_0 and r_0 and final states q_f and r_f respectively. Before each construction, we assume that the states q_f and r_f are first changed to nonfinal states, with each construction providing a new final state.



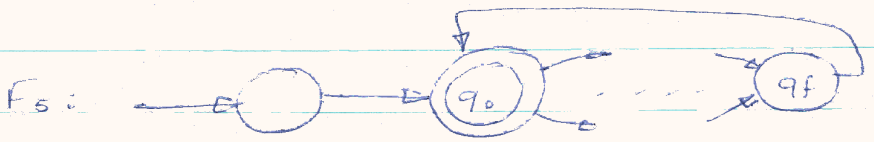
Atom: suppose R is the single symbol $a \in \Sigma$ or $a = \varepsilon$. Then this can be recognised by either:

Alternation: Assume that F_1 recognises R_1 and F_2 recognises R_2. Then we can construct F_3 to recognise $R_1 + R_2$:



Concatenation: We can construct F_4:

Closure: We can construct F_5



LIMITATIONS OF FSM's

Consider the set NMN, of all sequences of 0's and 1's consisting of a sequence of 1's, followed by a sequence of 0's and finally followed by another sequence of 1's the same length as the first. The set NMN can be written

$$NMN = \{ 1^n 0^m 1^n \mid n, m > 0 \}$$

Thm No FSR can recognise exactly the elements of NMN.

Proof Suppose M is an FSR recognising exactly NMN. Consider the set

$$S = \{ 1^n \mid 1 \leq n \leq k+1 \}$$

where k is the number of states of M. There must be at least two distinct members of S such that M is in the same state after reading either (as S has k+1 members but M has only k states). Say for example, these are 1^p and 1^q. Then, if we assume that M accepts $1^p 0^n 1^p$, M must also accept $1^p 0^n 1^q$, but this is not in NMN.

MODELS OF COMPUTATION & GRAMMARS

An effective procedure or algorithm is a "well-specified" and finite description of a computation that can be performed in a finite time.

We have seen that FSR's cannot exactly recognise NMN. We introduce push-down automata with a finite set Q of states, an input alphabet X and an infinite stack (all operations performed on top of stack). The initial and final states are as before. We also have the marker Z_0 at the TOS when we start.

Transitions look like $a, b \to w$

where a is the input, b is the symbol which must be on TOS for that transition, and w is a list of items to be pushed onto the stack. b is removed from the stack in the transition (replaced by w). If $w \neq \varepsilon$ then w has been pushed onto the stack, else the stack has been popped.

The operation of a PDA is described as a function: $f : Q \times X \times Y \to Q \times Y^*$

q_i a b — q_j NEXT(q_i, a, b) — stack string (q_j, a, b)

If the PDA is non-deterministic

$$f : Q \times X \times Y \to 2^{Q \times Y^*}$$

In a DPDA we associate to any triple exactly one next state and new stack string. All unmentioned arrows go to a reject state, thus eliminating confusion between null and non-null inputs.

Languages accepted by NDPDA's are called context free languages. No PDA can recognise a language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ because it can only count to two elements (up and down back to zero). However, this can be overcome by using more than one stack, or by an infinite tape with two directions (Turing machine)

There are languages which can be accepted by ~~a PDA~~ NDPDA's and not PDA's

TURING MACHINES

Church's Thesis: Any computation for which there is an effective procedure can be realised by a Turing machine.

We define Turing machines as having a finite set of states Q, an input alphabet X, and a work tape alphabet Y, as well as a set $F \subset Q$ of final states, plus an initial state.

Transitions are a mapping of state, inputs and current characters onto new states, new characters and directions. Once again, we may have both deterministic and non-deterministic machines. Both accept context sensitive type 0 languages. If they don't accept the input

they go into infinite loops and never halt.

Since DTM's are described by subsets of $Q \times X \times Y \times Q \times Y \times \{L, R, S\}$, and if $X = Y = \{0, 1\}$ and we have a fixed Q, then the number of possible DTM's is $2^{a \times \ldots}$, a countable amount.

The Universal Turing Machine is a TM which, given a description of any other TM, can emulate that TM. There is no TM however, which can decide whether any other program will halt or not (this is the halting problem - see proof in book).

GRAMMARS

We define language classes in terms of a set of restrictions on a general language definition scheme called production grammars.

BACKUS - NAUR FORM (Type 2 or context - free grammar)

Symbols

::=	is defined as
|	or
< >	used to enclose meta - variables

The LHS must be a single meta - variable
The collection of sentences obtained is called the language of the grammar. A language is ambiguous if it allows 2 essentially different derivations

CLASSES OF LANGUAGES.

There is a hierarchy of languages requiring more and more powerful grammars to describe them. Languages that can be described by RE's are called regular or type 3, those describable by BNF are called context free or type 2. Type 1 languages are called context sensitive. Type 0 languages are those that can be described by Turing Machines (most powerful), Type 2 languages correspond to NDPDA's

We now prove that type 3 languages can be described by BNF. Every R.E. R can be translated into the definition of a metavariable $\langle L_R \rangle$ describing the same language as follows.

atom If R is a single terminal symbol - say "a" - translate it to $\langle L_R \rangle ::= a$

alternation If $R = (R_1 + R_2)$, then
$\langle L_R \rangle ::= \langle L_{R_1} \rangle \mid \langle L_{R_2} \rangle$

composition If $R = (R_1 R_2)$ then
$\langle L_R \rangle ::= \langle L_{R_1} \rangle \langle L_{R_2} \rangle$

closure If $R = (R_1)^*$ then
$\langle L_R \rangle ::= \varepsilon \mid \langle L_{R_1} \rangle \langle L_R \rangle$

We can note some properties here:

DEFN A BNF grammar has extended right-linearity if its productions can be ordered so that for any production
$\langle A \rangle ::= \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$
each β_i contains only previously defined metavariables. Furthermore, if $\langle A \rangle$ appears in its own defn, it must occur as the last symbol in that β_i

THM Any extended right-linear BNF grammar describes a type 3 language. Any type 3 language has an extended right-linear BNF grammar.

(Similarly for left-linear grammars)

The power of BNF comes from what is called self-embedding. A BNF grammar is self embedding if there is at least one nonterminal of the language, say $\langle x \rangle$, such that $\langle x \rangle \rightarrow \alpha \langle x \rangle \beta$ where both α and β are nonempty strings. That is, starting with some metavariable, we can derive a sentential form in which that metavariable is surrounded by non-empty strings. Linear grammars preclude embedding which is why they are equivalent to type 3 (RE's) instead of type 2. If a context-free grammar is not self-embedding, it is regular.

PARSE TREES

First let us define a grammar for simple arithmetic expressions:

$\langle sae \rangle ::= \langle t \rangle \mid -\langle t \rangle \mid \langle sae \rangle + \langle t \rangle \mid \langle sae \rangle - \langle t \rangle$

$\langle t \rangle ::= \langle p \rangle \mid \langle t \rangle * \langle p \rangle \mid \langle t \rangle / \langle p \rangle$

$\langle p \rangle ::= X \mid (\langle sae \rangle)$

If we then take an $\langle sae \rangle$, we can show its structure in a parse tree. (see WHSF pg 362-3)

A BNF grammar is <u>ambiguous</u> if there is at least one sentence in the grammar for which there are two or more different parse trees. Equivalently, a grammar is ambiguous if there are two distinct derivations of at least one sentence. A consequence of the halting problem is that there is no TM that can determine whether an arbitrary BNF grammar is ambiguous or not.

The power of BNF comes from what is called self-embedding. A BNF grammar is self-embedding if there is at least one nonterminal of the language, say $\langle x \rangle$, such that $\langle x \rangle \rightarrow \alpha \langle x \rangle \beta$ where both α and β are nonempty strings. That is, starting with some metavariable, we can derive a sentential form in which that metavariable is surrounded by non-empty strings. Linear grammars preclude embedding which is why they are equivalent to type 3 (RE's) instead of type 2. If a context-free grammar is not self-embedding, it is regular.

PARSE TREES

First let us define a grammar for simple arithmetic expressions:

$\langle sae \rangle ::= \langle t \rangle \mid -\langle t \rangle \mid \langle sae \rangle + \langle t \rangle \mid \langle sae \rangle - \langle t \rangle$

$\langle t \rangle ::= \langle p \rangle \mid \langle t \rangle * \langle p \rangle \mid \langle t \rangle / \langle p \rangle$

$\langle p \rangle ::= X \mid (\langle sae \rangle)$

If we then take an $\langle sae \rangle$, we can show its structure in a parse tree. (see WHSF pg 362-3)

A BNF grammar is <u>ambiguous</u> if there is at least one sentence in the grammar for which there are two or more different parse trees. Equivalently, a grammar is ambiguous if there are two distinct derivations of a least one sentence. A consequence of the halting problem is that there is no TM that can determine whether an arbitrary BNF grammar is ambiguous or not.

Addendum

Sets & Relations

A *class* is a set of sets

A *family* or *collection* is a set of classes

The words subclass, subfamily, etc, are analogous to subset.

Thus the power set of a set, is the class of all subsets of the set

The word *space* means a non-empty set with a mathematical structure. The elements of a space are called *points*

Two sets are *disjoint* if they have no intersection

The product of the sets $A_1, \ldots A_n$ can be denoted by: $\prod_{i=1}^{m} A_i$. and consists of all m-tuples $\langle a_1, a_2, \ldots a_m \rangle$ where $a_i \in A_i$ for each $i = 1, \ldots, m$.

A relation from a set A to itself is called a relation in A.

The identity relation in any set A, denoted by Δ or Δ_A, is the set $\{\langle a, a \rangle \mid a \in A\}$. This is also called the *diagonal*, and is equivalent to $E(\Delta_A) = I_n$

If ρ is an equivalence relation in A, then the equivalence class of $a \in A$ is the set of elements to which a is related: $[a] = \{b \mid (a, b) \in \rho\}$.

The collection of equivalence classes of A, denoted by A/ρ, is called the quotient of A by ρ.

$$A/\rho = \{[a] \mid a \in A\}$$

The quotient set satisfies the following properties:

(i) $\forall a \in A,\ a \in [a]$

(ii) $[a] = [b]$ iff $\langle a, b \rangle \in \rho$

(iii) If $[a] \neq [b]$, then $[a]$ and $[b]$ are disjoint.

(iv) A/ρ is a partition of A. (by (i) $\Rightarrow$ (iii))

Functions

A subset f of $A \times B$ is a function iff each $a \in A$ appears as the first coordinate in exactly one ordered pair (a, b) in f.

If f is one-one and onto, and so has inverse function f^{-1}, then $f \circ f^{-1} = I_A$ and $f^{-1} \circ f = I_B$

Cardinality and Order

Two set A and B are equivalent, $A \sim B$, if there exists a function $f: A \to B$ which is one-one and onto. The function f is said to define a one-one correspondence between A and B.

A set is finite iff it is empty or equivalent to $\{1, 2, \ldots n\}$ for some $n \in N$; else it is infinite. Clearly, two finite sets are equivalent if they have the same number of elements. Note that by projection, the real numbers can be mapped onto the interval $(0, 1)$ or equivalently $[0, 1]$.

A set is <u>denumerable</u> if it has cardinality $\aleph_0$ (is equivalent to $\mathbb{N}$). If it is finite or denumerable it is <u>countable</u>

- Every infinite set contains a denumerable subset
- Every subset of a countable set is countable
- The union of a denumerable class of denumerable sets is also denumerable (sum for countable).

The unit interval $[0,1]$ is non-denumerable, and has cardinality c (the power of the continuum) $\sim \mathbb{R}$

<u>CANTOR'S THEOREM</u> The power set of any set A has cardinality greater than A

<u>CONTINUUM HYPOTHESIS</u> There does not exist a set A such that $\aleph_0 < \#(A) < c$.

A partial ordering is reflexive, transitive and anti-symmetric.
" " " is totally or linearly ordered if for every $a, b \in A$, <u>either</u> $a \leq b$ <u>or</u> $b \leq a$.

GROUPS

TRANSFORMATIONS Let M denote a finite or infinite collection of completely arbitrary objects (a set). If every element of M is associated with another well-defined element of the same set, we say a transformation is given. Every transformation of a finite set M can be given by means of a table consisting of two rows. In the upper we write the elements of M in an arbitrary order and below each of them we write the element corresponding. Eg $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$ or $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 3 \end{pmatrix}$

If some transformation of the set M is denoted by the letter A, then we denote by mA, where $m \in M$, the image of m under A. One-one transformations, particularly those of infinite sets, are also called permutations. Every one-one transformation has an inverse.

Consider a set A with a binary operation $\circ$. We say

$\circ$ is associative : $\forall a, b, c \in A, \quad (a \circ b) \circ c = a \circ (b \circ c)$

$\circ$ has identity : $\exists e \in A \ni \forall a \in A, \quad a \circ e = a = e \circ a$

$\circ$ has inverse : $\forall a \in A \ \exists a' \in A, \quad a \circ a' = a' \circ a = e$

$\circ$ is Abelian : $\forall a, b \in A, \quad a \circ b = b \circ a$ (commutative)

Semigroup : $\langle A, \circ \rangle$ where $\circ$ is associative

Monoid : semigroup plus identity property

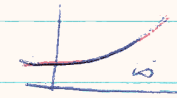
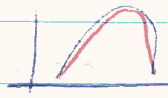
Group : Monoid plus inverse property

Abelian group : group plus abelian property

Field $\langle A, +, \circ \rangle \iff \begin{cases} \langle A, + \rangle \text{ is an abelian group} \\ \langle A - \{0\}, \circ \rangle \text{ is an abelian group} \\ \text{and distributive property holds} \end{cases}$

(additive inverse)

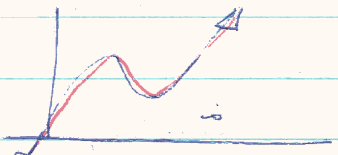
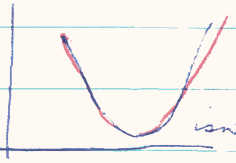
FUNCTIONS

injective
one - one — horizontal line test, is  is,  isn't

Has a left inverse

each horizontal line does not contain more than one point of f

surjective
onto —

Has a right inverse  is  isn't

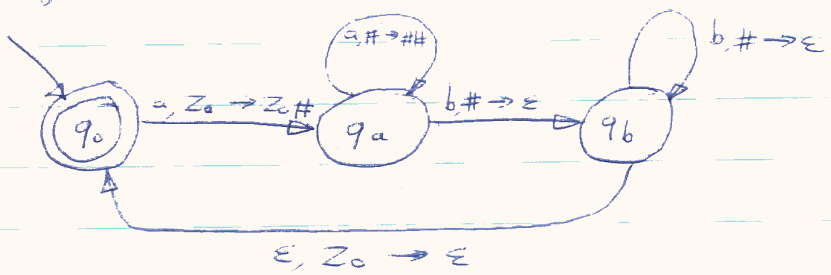
each horizontal line contains at least one point of f

PDA's & TM's

PDA's In the PDA, our labels on STD's are of the form $\langle$ input char, TOS char $\rangle \rightarrow \langle$ new TOS $\rangle$

The TOS char on the left is removed in the transition, and must be rewritten if it is still required. If input char = ε we have a null transition. If $\langle$ new TOS $\rangle = \varepsilon$ we simply drop the stack (only way of dropping stack).

Eg Recognises for $\{a^n b^n\}$



This loads a load onto the stack for each a, then removes them for each b. The set of symbols that may be pushed onto the stack is the stack alphabet, and must be finite.

By convention the stack is initialised to contain only Z_0. A string is rejected if the DPDA does not end up in a final state, if its stack empties (underflows), if it goes into an infinite loop of null transitions, or if it jams (no corresponding transition for input).

We adopt the convention that ? means any character. When it appears on both sides of the transition, it means 'whatever appeared on the input string'.

A configuration of a PDA consists of its state, the input pointer, and its stack. A trace of the configurations of our machine for the input string aaaabbbb is:

STATE	q_0	q_a	q_a	q_a	q_a	q_b	q_b	q_b	q_b	q_0
STACK					#					
				#	#	#				
			#	#	#	#	#			
		#	#	#	#	#	#	#		
	Z_0	Z_0	Z_0	Z_0	Z_0	Z_0	Z_0	Z_0	Z_0	
INPUT		a	a	a	a	b	b	b	b	

Note that many NDPDA's have no DPDA equivalent. Also, the language $\{a^n b^n c^n\}$ cannot be recognized by a PDA.

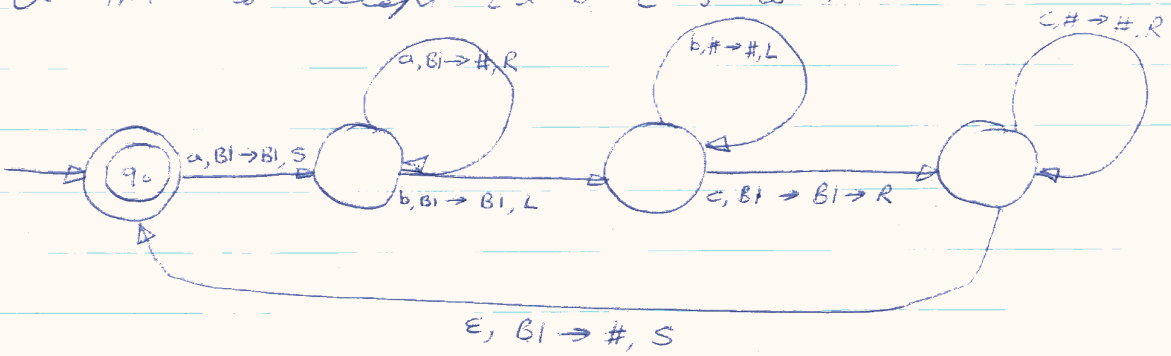
TURING MACHINES

A TM has a finite state control section, an input tape, and a seperate working tape. The working tape is unbounded on either side and contains blank symbols at the start. The STD arcs are labelled by pairs:

< input symbol, working symbol > → < output symbol, direction >

The symbol B1 stands for the blank work tape symbol. Nondeterminism adds no actual power (ie NDTM's have DTM equivalent).

A TM to accept $\{a^n b^n c^n\}$ is:



writing a hash over the blank is to prevent further transactions.

The halting problem states that no TM program can, given another program P, determine whether P will halt if P is given itself as input.

METALANGUAGES & PRODUCTIONS

When we are defining a language, the language employed in the definition is called a metalanguage (eg ~~RE's~~ are the language of RE's is the metalanguage for defining $(a)^* a$, $(ab)^* + c$, etc).

A powerful class of metalanguages is productio grammars. There are 3 types of symbols: terminals (symbols from the language being defined), metavariables or nonterminals (linguistic concepts from the language being defined) and metasymbols (punctuation used in the productions).

A string containing metavariables is called a sentential form, those with terminals only are sentences of the language, and the set of productions is called a grammar (use $< > \rightarrow |$)

Productn grammars can describe ~~class~~ type 0 languages (those recognised by TM's)

If we restrict the LHS of each production to single metavariable, & we have BNF (< > ::= |) (type 2)

Thm - For every NDPDA that recognises a language L, there is a BNF grammar for L
- For any language L' described in BNF, there is a NDPDA that accepts exactly L'.

RESIDUE ARITHMETIC

We define the relation $\equiv_p$ (called congruence mod p) on $\mathbb{Z}$ for each $p > 0$.

DEFN Let $A, B \in \mathbb{Z}$ and $p \in N^+$. Then:

$a \equiv_p b$ if $a - b = np$ for some $n \in \mathbb{Z}$

i.e., $(a-b)$ is a multiple of p.

We usually write $a \equiv_p b$ as $a \equiv b \pmod p$.

THM $\equiv_p$ is an equivalence rela on $\mathbb{Z}$ $(p \in N^+)$ prove

Rmk $a \equiv 0 \pmod p$ iff p divides a.

COR There are p equivalence classes $(\bmod p)$ namely $E(0), E(1), \ldots, E(p-1)$

These are called residue classes $(\bmod p)$ (prove)

THM Suppose $a \equiv b \pmod p$ and $c \equiv d \pmod p$

Then: (i) $a + c = b + d \pmod p$

(ii) $ac \equiv bd \pmod p$

(iii) $ag \equiv bg \pmod p$ for all $g \in \mathbb{Z}$.

From this we can deduce the 'rule of 9': A decimal number is divisible by 9 iff the sum of its digits is divisible by 9.

Thm 7 (Cancellation law)

If p is prime and $c \not\equiv 0 \pmod{p}$ (i.e. c is not a multiple of p)

then $ac \equiv bc \pmod{p} \Rightarrow a \equiv b \pmod{p}$

Cor (Fermat's Theorem)

Let p be prime and $\begin{cases} a \not\equiv 0 \pmod{p} \\ a \text{ not a multiple of } p \end{cases}$

Then $a^{p-1} \equiv 1 \pmod{p}$