Propositional Logic

What is Logic?

the laws of thought	(Boole, c1850, mathematics)
principles of right reasoning	(religion)
methodology of valid argumentation	(law)
study of grammar	(linguistics)
stages of cognitive development	(Piaget, developmental psychology)
truths based solely on the meaning of the terms	(math)
the most abstract and general description of reality	(philosophy)
force of reason rather than dogma	(politics)
science or history of the human mind	(Encyclopedia Britannica, 1771)
technique for design of	(computer science)

- circuitry
- program control
- process description
- structured programming
- deductive computation
- programming connectivity
- decision making in algorithms

Representation

Lexicon: the typographical forms which represent statements

Syntax: the rules of composition, making forms out of objects and functions.
Atomic objects are propositions, functions and relations.
Sentences are atomic objects + logical connectives.

Semantics: the rules of meaning, connecting statements to values

Boolean Algebra = Propositional Logic

Boolean algebra is the algebraic approach (match and substitute using equations), and *Propositional calculus* is the logical approach (inference using conjunction of facts) to the same mathematical structure, even though the fields developed independently, and don't talk to each other.

Both address the easiest and simplest useful formal system,
which poses the hardest and most important technical issues for computation.

Propositional Calculus

The simplest formal system with great utility. A proposition is the simplest complete unit of thought, any statement or decision with a Yes/No or True/False result.

Values: {True, False}

Objects: statements (propositions) that are either True or False {p, q, r, s...}

Operators: connectives {not, or, and, if-then, if-and-only-if}

Constructing Sentences

The logical connectives allow construction of compound ideas with several propositions.

For example: if (A and B) then (C or (not D))

The truth value of a compound sentence is the truth value of its component parts.

A and B	isTrue exactly when both A isTrue and B isTrue.
not A	isTrue exactly when A is False.
A or B	isTrue exactly when either A isTrue or B isTrue.
A implies B	isTrue exactly when either (not A) isTrue or B isTrue
A iff B	isTrue exactly when either A and B are both True or A and B are both False

There are 16 unique Boolean connectives of two variables, but only five are common {and, or, not, if-then, if-and-only-if}. All connectives can be expressed using only one {nor}.

Tautologies, Contradictions, and Indeterminate Sentences

Sentences that are always true regardless of the values of the atoms are called tautologies. A tautology conveys no information about its components.

Sentences that are always false are contradictions.

Sentences which do depend on (at least one of) their component atoms are indeterminate.

History of Logic

Ancient Party Games

Logic has confused, perplexed, and challenged philosophers and scholars from the beginning of culture. It was built into our language (and presumably our thinking) from the beginning of language. However, philosophers did not (and still do not) understand the subtleties of the simple words

{true, false, and, or, not, if, equal, some, all, therefore}

`Some men are barbarians.`
`Some barbarians are kind.`
`Thus, some men are kind.`
	Is this a proper conclusion?

`If it is raining, then I am happy.`	Is this necessarily True when I am in the rain?
`If I am dead, then I am happy`	Is this "if" the same as the above "if"?
`He or me.` `Watch or listen.`	Are there two types of "or"? (exclusive and inclusive)

`If you say that you are lying`
`and that is the truth,`
`then you are lying.`
	What do paradoxes mean? (Cicero)

`Is.`
`Not is.`
`Not not is.`
	Does “not not” mean nothing at all?

Aristotle

Aristotle was the first person to classify declarative language. He used three polar categories:

single vs compound	Socrates is happy vs Man is happy.
universal vs particular	Everyone vs someone.
affirm vs deny	Everyone vs no one.

The Syllogism

According to Aristotle, the fundamental unit of reasoning is the syllogism. He defined it as

> "discussion in which, when things are posited, other things necessarily follow."

```
All men are mortal.
Socrates is a man.
Thus, Socrates is mortal.
```

Syllogistic logic was developed into the first ever Axiomatic System with variables.

Scholastic Logic

The syllogism survived the Dark Ages in the form of the rules of theological debate. During the 13th century, Pope John XXI wrote a book on logic which dominated logical thought for the next 300 years. He observed that:

Nouns and Verbs   form   Subjects and Predicates

These subjects and predicates are CATEGORMATA; they have a referent in the real world.
The logical connectives are SYNCATEGOREMATA; they are without a referent in the real world.

Theological debates noticed the use/mention distinction:

`Man is mortal.`   versus   `Man is a noun.`

and the paradoxes generated by the absence of articles in Latin:

`The man is mortal.`   versus   `Man is mortal.`

Meanwhile in the Non-European World

In 10th century Baghdad, the Nestorian Abu Bishr Matta ibn Yunus refined Aristotle's logic, but his work was lost in the passage of time.

In India, logic was hotly debated in a form which differed only slightly from the syllogism:

`The mountain is fiery`	that is the Proposition
`Because smoky`	that is the Reason
`All that is smoky is fiery`	that is the Example
`So here`	that is the Application
`Therefore it is so.`	that is the Conclusion

The use of negation caused debate:

Why should the same words in different order have different meanings?

He shall-not look.

He shall not-look.

Not-he shall look.

"Absence of constant absence of pot is essentially identical with pot"
-- Mathuranatha c. 1700

In the West, Logic Evolved into Formal Systems

Renaissance:		Logic was ignored (experience was in vogue)
Enlightenment:		Leibniz sought a Universal Calculus of Reason, and studied Indistinguishability.
1850	Boole:	expressed sentences and noun expressions as algebra

x + y = y + x	associativity of OR
x (y + z) = x y + x z	distribution of AND over OR
if x = y then x + z = y + z	algebraic substitution

1880	Venn:	logical diagrams
1885	Peirce:	truth tables
1900	Russell:	logical foundations of mathematics
1920	Post:	metalogic (just what are we doing?)

Crisis in the Twentieth Century

Oh No! There is no consistency in mathematics, there are paradoxes in every system.

Logicism	Bertrand Russell	Mathematics is identical to logic. (We'll patch the holes.)
Intuitionism	L.E. Brouwer	Mathematics presupposes concepts. Concepts rest on natural numbers. (We'll construct what is known, and not admit infinity.)
Formalism	David Hilbert	Mathematics is a set of syntactic transformations. (We'll refuse to interpret it.)

Logical Tautologies

1.	$P \vee \neg P$	excluded middle
2.	$\neg(P \wedge \neg P)$	noncontradiction
3.	$\neg\neg P = P$	double negation
4.	$(P \wedge Q) \rightarrow P$	simplification
5.	$P \rightarrow (P \vee Q)$	simplification
6.	$(P \wedge P) = P$	idempotence
7.	$(P \vee P) = P$	idempotence
8.	$(P \wedge (P \rightarrow Q)) \rightarrow Q$	modus ponens
9.	$((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R)$	syllogism
10.	$(P \rightarrow Q) = (\neg Q \rightarrow \neg P)$	contraposition
11.	$((P \rightarrow Q) \wedge \neg Q) \rightarrow \neg P$	modus tollens
12.	$((P \vee Q) \wedge \neg P) \rightarrow Q$	disjunctive syllogism
13.	$(P \rightarrow Q) = (\neg P \vee Q)$	conditional disjunction
14.	$(\neg P \rightarrow (Q \wedge \neg Q)) \rightarrow P$	reductio ad absurdum
15.	$(((P \rightarrow R) \wedge (Q \rightarrow S)) \wedge (P \vee Q)) \rightarrow (R \vee S)$	dilemma
16.	$(P \rightarrow (Q \rightarrow R)) = ((P \wedge Q) \rightarrow R)$	exportation
17.	$(P = Q) = ((P \rightarrow Q) \wedge (Q \rightarrow P))$	biconditional
18.	$\neg(P \vee Q) = (\neg P \wedge \neg Q)$	DeMorgan
19.	$\neg(P \wedge Q) = (\neg P \vee \neg Q)$	DeMorgan
20.	$\neg(P \rightarrow Q) = (P \wedge \neg Q)$	negation of conditional

21.	$\neg(P = Q) = (\neg P = Q)$	negation of biconditional
22.	$(P \vee Q) = (Q \vee P)$	commutativity
23.	$(P \wedge Q) = (Q \wedge P)$	commutativity
24.	$(P = Q) = (Q = P)$	commutativity
25.	$((P \vee Q) \vee R) = (P \vee (Q \vee R))$	associativity
26.	$((P \wedge Q) \wedge R) = (P \wedge (Q \wedge R))$	associativity
27.	$((P = Q) = R) = (P = (Q = R))$	associativity
28.	$(P \wedge (Q \vee R)) = ((P \wedge Q) \vee (P \wedge R))$	distribution
29.	$(P \vee (Q \wedge R)) = ((P \vee Q) \wedge (P \vee R))$	distribution
30.	$(P \rightarrow (Q \vee R)) = ((P \rightarrow Q) \vee (P \rightarrow R))$	distribution
31.	$(P \rightarrow (Q \wedge R)) = ((P \rightarrow Q) \wedge (P \rightarrow R))$	distribution
32.	$((P \vee Q) \rightarrow R) = ((P \rightarrow R) \wedge (Q \rightarrow R))$	disjunction/conditional
33.	$((P \wedge Q) \rightarrow R) = ((P \rightarrow R) \vee (Q \rightarrow R))$	conjunction/conditional
34.	$(P \rightarrow Q) \rightarrow ((R \wedge P) \rightarrow (R \wedge Q))$	factorization
35.	$(P \rightarrow Q) \rightarrow ((R \vee P) \rightarrow (R \vee Q))$	summation

Logical Proof

Ways of Expressing the Mathematics of Logic

Boolean connectives (and, or, not, if-then, if-and-only-if)
function tables (truth tables)
Boolean algebra
Venn diagrams
switching circuits
transistor arrays (silicon chips)
Boolean lattice
Boolean cubes (blocks in space)
matrix logic
boundary logic

Ways of Computing the Mathematics of Logic

exhaustive listing of possibilities	(truth tables)
deduction/inference	(Boolean connectives)
algebra	(Boolean algebra)
spatial overlap	(Venn diagrams)
current through transistors	(circuitry)
partial orderings	(lattices)
spatial conjunction	(cubes)
operators	(matrix logic)
containment	(boundary logic)

Mechanisms of Proof

Truth tables	
Natural deduction	
Resolution	(not covered in class)
Boundary logic	
Induction	

Truth Table Analysis

Examining all the possibilities is exponential: there are 2^n cases to evaluate for n variables even in the simplest case of propositional logic. However, lookup tables are a brute force method that is easy to understand. The technique is to list all possible combinations of values for each variable, and use simple definitions of the logical connectives to evaluate compound sub-expressions.

Example: $(P \land Q) \rightarrow (R = \neg S)$

P	Q	R	S	¬S	P ∧ Q	R = ¬S	P∧Q → R=¬S
T	T	T	T	F	T	F	F
T	T	T	F	T	T	T	T
T	T	F	T	F	T	T	T
T	T	F	F	T	T	F	F
T	F	T	T	F	F	F	T
T	F	T	F	T	F	T	T
T	F	F	T	F	F	T	T
T	F	F	F	T	F	F	T
F	T	T	T	F	F	F	T
F	T	T	F	T	F	T	T
F	T	F	T	F	F	T	T
F	T	F	F	T	F	F	T
F	F	T	T	F	F	F	T
F	F	T	F	T	F	T	T
F	F	F	T	F	F	T	T
F	F	F	F	T	F	F	T

Deduction

The rules of inference, or natural deduction, apply at three different levels of abstraction: individual propositions, individual sentences, and collections of sentences. Modus Ponens serves as an example.

Atoms: $(p \land (p \rightarrow q)) \rightarrow q$
Sentences: $(A \land (A \rightarrow B)) \rightarrow B$
Sets of sentences: $(\{A,B...\} \land (\{A,B...\} \rightarrow \{C,D...\})) \rightarrow \{C,D...\}$

Natural Deduction Proof Techniques

Natural deduction evolved from natural language and from human intuition, so it is relatively easy to understand. It is very difficult to find the right rules to apply at the right time. Humankind has had an extremely difficult time coming to understand logic, and logic itself is still undergoing extreme revision. Below, |= means "this follows logically":

Modus Ponens: $A \land (A \rightarrow B) \quad \models \quad B$

Modus Tollens: $\neg B \land (A \rightarrow B) \quad \models \quad \neg A$

Dilemma: $(\neg A \lor B)\land(A \rightarrow C)\land(B \rightarrow C) \quad \models \quad C$

Contradiction: $(A \rightarrow B) \land \neg B \quad \models \quad \neg A$

Natural Deduction Example

Premise 1:	If he is lying, then (if we can't find the gun, then he'll get away).
Premise 2:	If he gets away, then (if he is drunk or not careful, then we can find the gun).
Premise 3	It is not the case that (if he has a car, then we can find the gun).
Conclusion:	It is not the case that he is both lying and drunk.

Encode the propositions as letters:

L = he is lying	G = we can find the gun	A = he will get away
D = he is drunk	C = he is careful	H = he has a car

Premise 1:	if L then (if (not G) A)
Premise 2:	if A then (if (D or not C) then G)
Premise 3:	not (if H then G)
Conclusion:	not (L and D)

Encode the propositions using logical connectives:

P1:	$L \rightarrow (\neg G \rightarrow A)$
P2:	$A \rightarrow ((D \vee \neg C) \rightarrow G)$
P3:	$\neg(H \rightarrow G)$
C:	$\neg(L \wedge D)$

Figure out a good proof strategy. This step is the source of difficulty in natural deduction. In the Contradiction Strategy, we assume the negation of the conclusion and plan to show a contradiction:

1.	$(L \wedge D)$	contradiction of the conclusion
2.	L	simplification of 1
3.	D	simplification of 1
4.	$\neg G \rightarrow A$	modus ponens with 2 and P1
5.	$\neg(\neg H \vee G)$	rewrite P3 with conditional exchange: $X \rightarrow Y = \neg X \vee Y$
6.	$\neg(\neg H \vee \neg\neg G)$	double negation of part of 5
7.	$H \wedge \neg G$	rewrite 6 with DeMorgan: $\neg(\neg X \vee \neg Y) = X \wedge Y$
8.	$\neg G$	simplification of 8
9.	A	modus ponens with 8 and 4
10.	$((D \vee \neg C) \rightarrow G)$	modus ponens with 9 and P2
11.	$(D \vee \neg C)$	addition of $\neg C$ to 3
12.	G	modus ponens with 11 and 10
13.	$G \wedge \neg G$	conjunction of 8 and 12
14.	$\neg(L \wedge D)$	steps 1 to 13 have created a contradiction: $G \wedge \neg G$ = False, so the negation of the conclusion on line 1 is False. Therefore the conclusion must be True.

Boundary Logic

Advances in knowledge must necessarily appear to be unintelligible before their discovery and simple or obvious after their discovery.

Challenge

Computation and logic (Boolean algebra) are universally built on binary representations.

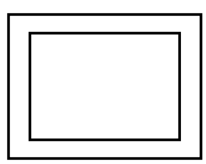
`0 1` `True False` `Yes No`

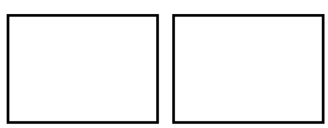
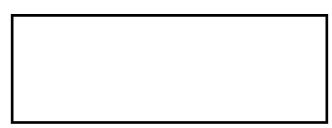
Is there a simpler approach? Can logic be expressed in a unary notation?

Boundary Mathematics

The use of delimiting tokens, or containers, as both constants and functions. Here is an (pure math) example:

`Common boundaries cancel.`

 = `<void>`

 = 

Concepts

Boundary Token	an enclosure
Representational Space	the bounded space

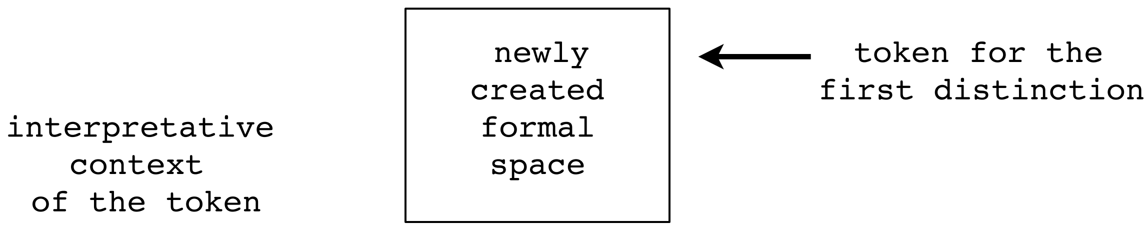
Two Voids

Absolute void	that which cannot be referred to without contradiction
Relative void	emptiness enclosed within a boundary

Constructing a Distinction

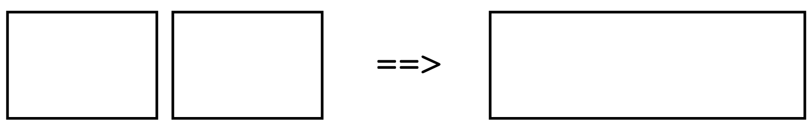
A `Universal Distinction` is first boundary we agree upon. In forming a first distinction, we construct three things simultaneously:

a formal space (inside)
a token representing the distinction (boundary)
a context from which to interpret the distinction (outside)



Calling

Focus your attention on the outside, where you see the mark (the usual viewing point).
Call the boundary that you see a "symbol".

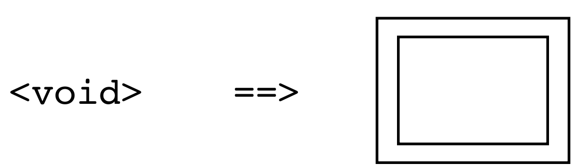


`To call is to maintain perspective.`

Calling is the rule of invariance. It is also is the rule of naming. Thus the relationship between an object and its name is invariant.

Crossing

Focus your attention on the inside of a mark, where there is empty space.
Cross the boundary to the outside. Now you can see a mark.



`To cross is to change perspective.`

Crossing is the rule of variance. It is also a process of changing.

The Arithmetic of Boundaries

```
CALLING        ( ) ( )  =  ( )

CROSSING       ( ( ) )  =
```

Moving to Algebra

The ground values of boundary logic are one token { ( ) } and the absence of that token. If an equation holds for all ground values, it holds in general. Using this, we can construct algebraic truths from the cases of the arithmetic:

```
        DOMINION            INVOLUTION            PERVASION

      ( ) ( ) = ( )       (( ( ) )) = ( )       ( ( ) ) ( ) = ( )
      ( )     = ( )       ((     )) =           (     )     = ( )
thus
      ( )  A  = ( )       ((  A  )) =  A        (  A  )  A  = ( )
```

Boundary Logic Rules of Transformation

The transformation axioms of boundary logic are:

Dominion (the halting condition, when to stop)

```
( ) A = ( )                      REIFY   <==>  ABSORB
```

Involution (double negation, how to remove excess boundaries)

```
((A)) = A                        ENFOLD  <==>  CLARIFY
```

Pervasion (how to remove excess replications of variables)

```
A (A B) = A (B)                  INSERT  <==>  EXTRACT
```

Each axiom suggests the same strategy for computation: `erase irrelevant structure`

Algebra provides the useful tool of substitution. Any transform can be applied at any time and at any place in the expression without changing the value of the expression. Thus, any transformation path does not change the value of an expression. It doesn't matter how you get to a simpler expression (an answer). Some paths may be longer and less efficient, but all lead to equivalent results.

Boundary Logic

Boundary logic uses a spatial representation of the logical connectives. Since CALLING provides an object-oriented interpretation, and CROSSING provides a process-oriented interpretation of the same mark, boundary forms can be evaluated using either an algebraic (match and substitute) approach or a functional (input converted to output) process.

Representation of logic and proof in spatial boundaries is new, and quite unfamiliar. Boundary logic is not based on language or on typographical strings, nor is it based on sequential steps. Boundary techniques are inherently parallel and positional. The meaning, or interpretation, of a boundary form depends on where the observer is situated. From the outside, boundaries are objects. From the inside, you cross a boundary to get to the outside; boundaries then are processes. This dramatically different approach (that is, permitting the observer to be an operator in the system) does not change the logical consequences or the deductive validity of a logical process.

Spatial representations do not have the concepts of associativity and commutativity. The base case is no representation at all, that is, the void has meaning in boundary logic. Simplification of logical expressions occurs by erasure of irrelevancies rather than by accumulation of facts.

Boundary Logic Representation

LOGIC	BOUNDARY	COMMENTS
False	<void>	No representation. Note: (( )) = <void>
True	( )	The empty boundary
A	A	Objects are labeled by tokens
not A	(A)	Negation is on the other side
A or B	A B	Disjunction is sharing the same space
A and B	((A)(B))	Conjunction is a special configuration
if A then B	(A) B	Implication is separation by a boundary
A iff B	(A B)((A)(B))	Equality is spatial complex

In the above map from conventional logic to boundaries, the many textual forms of logical connectives condense into one boundary form. Note that the parens, ( ), is a linear, or one-dimensional, representation of a boundary. Circles and spheres are expressions of boundaries in higher dimensions.

Boundary Logic Examples of Proof

```
To Prove                       Transcribe and Apply the Three Axioms

A → A                              (A) A
                                   ( ) A            pervasion
                                   ( )              dominion

¬¬A = A                            ((A)) = A
                                     A   = A        involution

((A → B) ∧ A) → B              (( (A) ((A) B) ))  B
                                  (A) ((A) B)     B    involution
                                  (A) (     )     B    pervasion of B and (A)
                                      (     )          dominion

A ∧ B = ¬(¬A ∨ ¬B))            ((A)(B)) = ((A)(B))     identity
```

The Fruit Problem

Premise 1: If A then (if (not P) C)
Premise 2: If C then (if (O or not K) then P)
Premise 3: Not (if B then P)
Conclusion: Not (A and O)

Encode the logical connectives as boundaries, and simplify:

```
P1:     (A) ((P)) C      ==>   (A)  P  C
P2:     (C) (O (K)) P
P3:     ( (B) P )
C:      ( ((A) (O)) )    ==>   (A)  (O)
```

Join all premises and conclusions into one form, using (P1 ∧ P2 ∧ P3) → C

```
( ((P1) (P2) (P3)) ) C  ==>  (P1) (P2) (P3) C     involution
```

Substitute the forms of the premises and conclusion, and reduce:

```
( (A) P C )  ( (C) (O (K)) P )  ( ((B) P) )  (A) (O)
( (A) P C )  ( (C) (O (K)) P )     (B) P     (A) (O)    involution
( (A)   C )  ( (C) (O (K))   )     (B) P     (A) (O)    pervasion of P
(       C )  ( (C) (O (K))   )     (B) P     (A) (O)    pervasion of (A)
(       C )  (     (O (K))   )     (B) P     (A) (O)    pervasion of (C)
(       C )         O (K)          (B) P     (A) (O)    involution
(       C )         O (K)          (B) P     (A) ( )    pervasion of O
                                                 ( )    dominion
```

The Age of Mathematical Concepts and Symbols

Our clarity of understanding of mathematical concepts corresponds to the time evolution of these concepts. That is, older is simpler. As well, the sequence of math concepts taught in schools pretty much follows the historical evolution of mathematical ideas. Here is a rough road map of the time evolution of various mathematical concepts. Asterisks, *, mark content covered in class.

8000 BC*		one-to-one correspondence
4000 BC*	1,2,3...	counting
1000 BC	.	zero (as dot)
400 BC*		zero as blank space
300 BC*	0	zero
300 BC*		syllogistic logic
1050	––	horizontal fraction bar
1417*	+	plus
1425	%	percent
1432*		mathematician
1484*	a^n	exponent
1484*		billion, trillion,...
1530	0.0	decimal fractions
1544		division
1549		parallel
1551*		irrational numbers
1551		theorem
1556*	()	parentheses
1557*	=	equals
1570*	A = B	equation
1570*	2,3,5,7...	prime number
1575*	x	variables as letters
1583	sin	sine function
1618	*	times (X in 1618, * in 1659)
1624	log	logarithm function
1631	>	greater/less than
1634		angle
1637*		imaginary, real (Descartes)
1647*	π	pi
1655	A,B,C	lettering for triangles
1655*	∞	infinity
1672*		“math” (Newton)

1674	cos	cosine function
1675	d/dx	derivative, integral
1690	e	base of natural logs
1718*		probability
1734*	f(x)	function symbol
1763*		natural number
1770	∂	partial derivative
1777*	i	imaginary unit
1786	lim	limit
1808*	!	factorial
1816*	ax = bx+c	linear equation
1827		long division
1839		“Fermat’s last theorem”
1840		pencil
1841	l a l	absolute value
1843	[]	matrices
1848	(x+a)(x+b)	factor
1851*	{a,b,c}	set
1882		isomorphism
1883		eigenvalue
1887		tensor
1888*	U	union, intersection
1891		histogram
1892		standard deviation
1902*	e	identity element
1910*	~, V	symbols for not, or, and
1921*		truth table
1931		spinor
1935		homomorphism
1938*	10^{100}	googol, googolplex
1940*	Ø	null set
1940		onto
1975*		fractal
1975		chaos
1989*	(( ))	boundary mathematics

Induction and Recursive Definitions

Almost all mathematical structures are defined by induction (recursion). An inductive definition consists of three components:

- a base case, the simplest possible application of the induction
- an inductive case which assumes an arbitrary member of the domain (all possible objects), and constructs the adjacent member.
- an ordering principle which provides a structure for inferring that when one member can be constructed from adjacent member, then all members can be constructed.

Mathematical Induction

The idea is to demonstrate truth for the base case (the simplest member of the ordered set), and then to demonstrate the truth for an arbitrary member of the set, assuming the truth of the member next to it in the order relation.

```
If N is an ordered set and property P isTrue for
      1) the minimal member of N, and
      2) if P(x) then P(next(x)) for an arbitrary member x,
then P isTrue for all members x of N.
```

Using the natural numbers, `N = {1, 2,...}`:

```
If P(1) isTrue, and
      assuming P(x) we can show that P(x+1) isTrue, then
P(x) isTrue for all natural numbers.
```

Examples

Domain	Base	Inductive Step
integers	0	f[n] –> f[n+1]
sets	{ }	f[S] -> f[S•u]
lists	empty list	P[first] -> P[next]
parentheses	( )	x -> (x) x and y –> xy

The Integer Domain

```
Notation:          n+1 = Successor[n] = n'

Counting:          not[x' = 0]
                   1      = 0'
                   n + 1  = n'

Addition:          m + 0  = m
                   m + n' = (m + n)'

Multiplication:    m * 0  = 0
                   m * n' = (m * n) + n

Exponentiation:    m ^ 0  = 1
                   m ^ n' = (m ^ n) * n

Summation:         sum[0]  = 0
                   sum[i'] = sum[i] + i'

Factorial:         fac[0]  = 1
                   fac[i'] = fac[i] * i'

Power-of-Two:      power-of-2[0]  = 1
                   power-of-2[n'] = 2*power-of-2[n]
```

Using mathematical induction, prove the following for integers:

Rule of Distribution: `(i*j) + (i*k) = i*(j+k)`

```
   i=0:  (0*j)      + (0*k)      = 0*(j+k)
 any i:  (i*j)      + (i*k)      = i*(j+k)
         (i*j) + j + (i*k) + k = i*(j+k) + j + k       add
         (i+1)*j    + (i+1)*k    = i*(j+k) + 1*(j+k)   definition of +
         (i+1)*j    + (i+1)*k    = (i+1)*(j+k)         inductive step
           i' *j    +   i' *k    =   i' *(j+k)         definition of '
```

Algebraic Summation: `2*sum[n] = n*(n+1)`

```
   n=0:  2*  0                 = 0*(0+1)
 any n:  2*sum[n]              = n*(n+1)
         2*sum[n] + 2(n+1)     = n*(n+1) + 2(n+1)      add
         2*(sum[n] + (n+1))    = (n+2)*(n+1)           distribution
         2*sum[n+1]            = (n+1)*(n+2)           definition of sum
         2*sum[n']             = (n')*(n'+1)           definition of '
```

Functions

Ordered Pairs

A function is specified by a collection of ordered pairs, `(a,b)`. The members of the ordered pair are elements, or members, of a set.

Example:

The integer double function `2a` is defined by an (infinite) collection of ordered pairs of the form `(a,b)`, where the values of `a,b` are in the set of `integers`:

```
2a =def= {(0,0), (1,2), (2,4), (3,6), ...}
```

Functions and Relations

relation: `xRy isTrue` function: `f(x)=y isTrue`

The set of all first values of a set of ordered pairs is called the Domain.

The set of all second values of a set of ordered pairs is called the Range.

A relation is a collection of ordered pairs over two sets, the domain set and the range set.

A function is a relation `(x,f(x))`, such that

1. Every member of the domain is associated with a member of the range, and
2. No element in the domain is associated with more than one element in the range.

Perspectives on Functions

1. Formal constraints on a relation

existence: `all x inDomain . exists y inRange`

uniqueness: `all pairs (x,f(x)) . if x1=x2 then f(x1)=f(x2)`

2. Graph

Domain on x-axis, Range on y-axis
uniqueness permits the graph to cross any vertical line (i.e. x-value) only once.

3. Lookup table

```
x     f(x)
1     1
2     4
3     9
```

4. Static relation between variables

`x = y + 5` "=" is an equivalence relation

5. Dynamic relation between variables

`f(x) = y` `x` is the independent variable (controlled measurement)
`y` is the dependent variable (observed measurement)

6. Rule of correspondence/algorithm

take a number	`x`
double it	`2*x`
add 3	`2*x + 3`

7. Set transformation

```
Domain        Range
  a   ----------->   b
  b   ----------->   c
  c   ----------->   d
  d   ----------->   d
```

8. Input-output machine

```
     x
 \       /
  \     /
    |  |
    |  |

    f(x)
```

9. Way of finding and assigning names to unnamed objects

`2^100` is the short name of a large number

10. Directed graph

```
(1) ---> (3) ---> (5)
```

Types of Functions

Surjective, Onto, Epic	all y inRange, exists x inDomain . f(x) = y
Injective, 1-to-1, Monic	if f(x1) = f(x2) then x1 = x2
Bijective	1-to-1 and Onto

Bijective functions have an inverse, since every element in both the Domain and the Range are in correspondence:

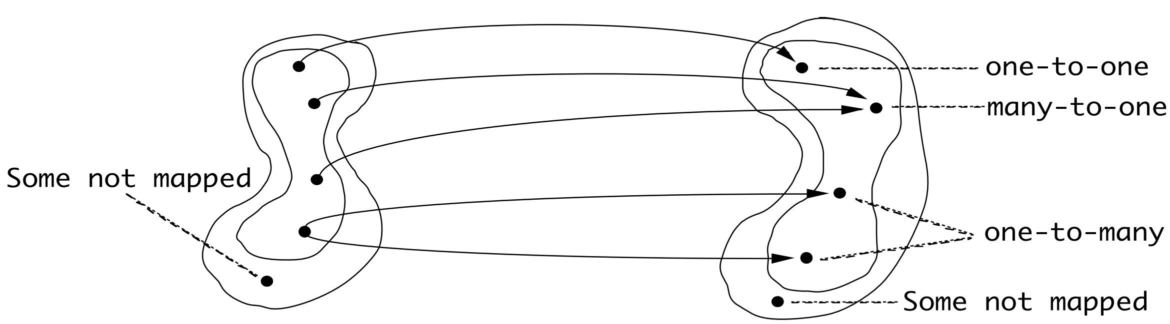
two-way existence	all x inD, exists y inR . f(x) = y
	all y inR, exists x inD . f(x) = y
two-way uniqueness	all (x,f(x)) . x1 = x2 iff f(x1) = f(x2)
inverse:	Exists f-inverse iff f is onto and one-to-one

Special Functions

Identity	f(x) = x
Characteristic	f(x) = 1 if x inA
	= 0 if x not inA
Permutations	(1,2,3) <--> (3,1,2) <--> (2,3,1)
Sequences	1 .. n <--> 1/1 .. 1/n

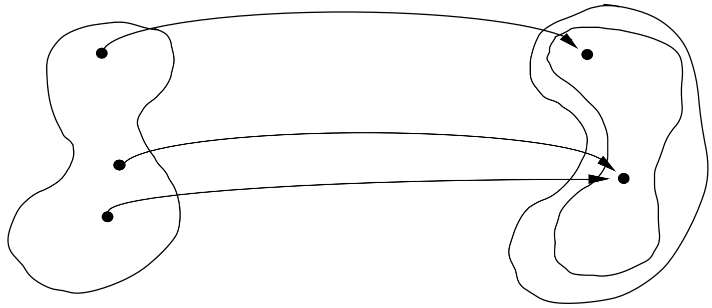
Mappings

===Relation===



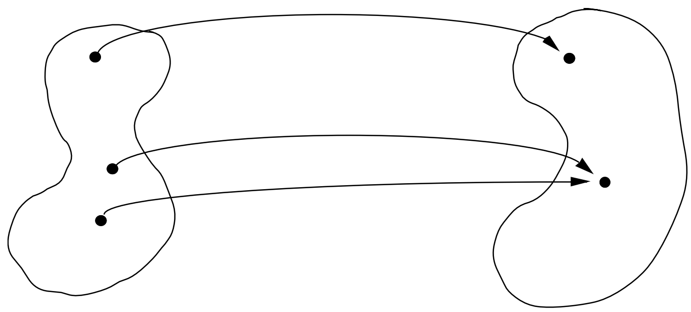
===Function===

EXISTENCE
=def=
all mapped



UNIQUENESS
=def=
no one-to-many

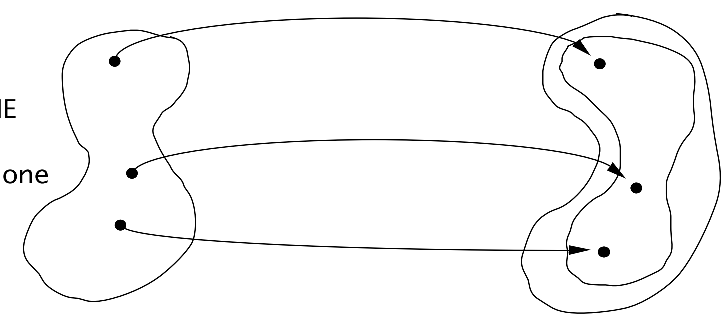
===Surjective/Onto/Epic Function===



ONTO
=def=
all mapped

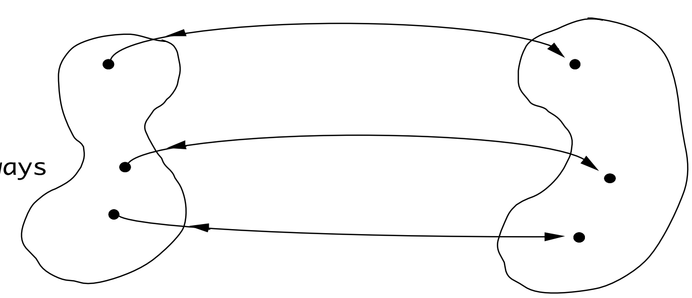
===Injective/1-to-1/Monic Function===

ONE-TO-ONE
=def=
no many-to-one



===Bijective/1-to-1 and Onto Function===

INVERSE
=def=
maps both ways



ONE-To-ONE and ONTO
=def=
no many-to-one and
all mapped in both
Domain and Range

Function Composition

`(f o g) = ` All pairs `(x,z)` Exists `y` such that `(x,y)` in `g` and `(y,z)` in `f`
Note that the Range of g is a subset of the Domain of f

`(f o g)(x) = f( g(x) )`

Associative: `(f o g) o h = f o (g o h)`

Not commutative: `f o g =/= g o f`

Maintains the type of the function:

if `f` and `g` are functions, then `(f o g)` is a function
if `f` and `g` are onto, then `(f o g)` is onto
if `f` and `g` are one-to-one, then `(f o g)` is one-to-one

Composition of a function with its inverse:

`f o f-inverse = identity I` on Range of `f`
`f-inverse o f = identity I` on Domain of `f`

Inverse of a composition: `(f o g)-inverse = g-inverse o f-inverse`

Binary Functions

Binary functions are a mapping of ordered pairs onto elements: `((a,b) c)`

e.g.: `a + b = c` `+ = {((a,b),c) such that (a,b) in S X S and c inS}`

The domain consists of ordered pairs rather than single elements.

If `a`,`b`, and `c` are in the Domain,
then the Domain is closed with regard to the function:

`All x1,x2 inD such that f(x1,x2) inD`

Algebraic Systems

An *algebraic system* consists of:

— a collection of names (labels)
— an operation that connects names with other names
— rules for building expressions and equations out of names and operations
— rules that permit changing expressions without changing what they mean

Here is an example, the algebraic system for *addition of whole numbers*:

	Components	**Examples**
Names	whole numbers	`1, 2, 3, 24`
Operation	addition (+)	`1 + 2 = 3,   3 + 7 = 10`
Expressions	`a + b`	`1 + 2,  x + 5,  x + y + z`
Rules	`a + b = b + a`	`1 + 2 = 2 + 1`
	`a + (b + c) = (a + b) + c`	`1 + (2 + 3) = (1 + 2) + 3`
	`a + 0 = a`	`5 + 0 = 5`

Here is an algebraic system you have never seen before, *putting letters inside or outside*:

	Components	**Examples**
Names	letters	`x, y, z`
Operation	inside or outside	`(x),   x (y)`
Expressions	`a (b)`	`(x y), ((x)) y, ((x y))`
Rules	`(a)(b) = (a b)`	`(x y)(z z) = (x y z z)`
	`a a = (a)`	`x y z x y z = (x y z)`

Algebraic Systems -- Problem Solving

Here are the two rules of the *Inside/Outside system*:

`(a)(b) = (a b)`   `a a = (a)`

Which of the following are correct equations in this system?

A. `x (y z) = (x y) z`

B. `w w w w = ((w))`

C. `(m (n)) = ((m) n)`

PROBLEM SOLVING STEPS: Answer each question very briefly (less than 10 words each).

1. What is the problem?

2. What do you already know?

3. What will you do to find the answers? What skills or tools can you use?

4. Find the answers. Show your work.

5. How confident are you about your answers?

6. What did you learn about algebraic systems?