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 possibilities is exponential in number of cases: there are 2^n cases to evaluate for n variables even in the simplest case of propositional logic without functions or relations. However, lookup tables are a brute force algorithm that is easy to understand and to implement. 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: `if (P and Q) then (R = (not S))`

P	Q	R	S	(not S)	(P and Q)	(R = (not S))	(if P&Q then 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 and (p -> q)) -> q`

Sentences: `(A and (A -> B)) -> B`

Sets of sentences: `({A,B...} and ({A,B...} -> {C,D...})) -> {C,D...}`

Deductive Steps

There are three separate concepts of **proof step** (written above as "implies") which have been shown to be equivalent:

material implication, logical implication, and computation.

Material implication: p -> q

is defined by the Truth Table of values. Notice that "(if False then True) is True", (the second row) does not make sense in language structures, it is True by definition.

p	q	(p -> q)
0	0	1
0	1	1
1	0	0
1	1	1

Logical Implication: p |= q

is defined by common sense and by the rules of deduction.

"if (p is logically True) then (q is logically True)"

Formal Proof: p |- q

is defined by taking logical implication steps from p to q

"if (p is True) then a sequence of implications shows (q is True)"

The Rules of Natural Deduction

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 (exponential in difficulty of use). Recall that humankind has had an extremely difficult time coming to understand logic, and logic itself is still undergoing extreme revision. "**|=**" means "logically implies" while "**->**" is simply a symbol referring to a particular truth table. The same subtle difference exists between "**and**" and "**&**".

Modus Ponens:	A and A -> B	|=	B
Modus Tollens:	~B and A -> B	|=	~A
Double negation:	A	|=	~~A
	~~A	|=	A
Conjunction:	A and B	|=	A & B
Simplification:	A & B	|=	A
	A & B	|=	B
Addition:	A	|=	A v B

Natural Deduction Proof Techniques

Modus Ponens:	A and A |- B	|=	B
Modus Tollens:	~B and A |- B	|=	~A
Conditional proof:	A |- B	|=	A -> B
Dilemma:	(A or B) and (A |- C) and (B |- C)	|=	C
Contradiction:	(A |- B) and ~B	|=	~A
Cases:	(A is True |- B) and (A is False |- B)	|=	B

Note that "|-" is a sequence of formal steps, while "|=" is assurance of logical truth.

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 logical connectives:

```
P1:   L ->  (~G ->  A)
P2:   A ->  ((D v ~C) -> G)
P3:   ~(H -> G)
C:    ~(L & D)
```

Figure out a good proof strategy. This step is not algorithmic, and is the source of difficulty in natural deduction approaches. Here the Contradiction strategy works:

1.	(L & D)	assume the negation of the conclusion, and plan to show a contradiction
2.	L	simplification of 1
3.	D	simplification of 1
4.	~G -> A	modus ponens with 2 and P1
5.	~(~H v G)	rewrite P3 with conditional exchange: X -> Y = ~X v Y
6.	~(~H v ~~G)	double negation of part of 5
7.	H & ~G	rewrite 6 with DeMorgan: ~(~X v ~Y) = X & Y
8.	~G	simplification of 8
9.	A	modus ponens with 8 and 4
10.	((D v ~C) -> G)	modus ponens with 9 and P2
11.	(D v ~C)	addition of ~C to 3
12.	G	modus ponens with 11 and 10
13.	G & ~G	conjunction of 8 and 12
14.	~(L & D)	steps from 1 to 13 have created a contradiction: G & ~G = False, therefore the assumption on line 1 is false. But that assumption is the negation of the conclusion. Therefore the negation of the negation of the conclusion is True. That is, the conclusion is True.