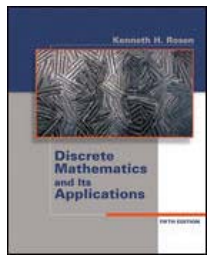
Ch.1 (Part 1): The Foundations: Logic and Proof, Sets, and Functions

- Introduction
- Logic (Section 1.1)



Introduction

- Formal Languages (computer languages)
- Compiler Design
- Data Structures
- Computability
- Automata Theory
- Algorithm Design
- Relational Database Theory
- Complexity Theory (counting)

Example (counting):

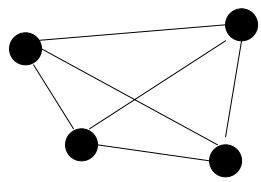
- The Traveling Salesman Problem

Important in
- circuit design
- many other CS problems

_____

Given:
- n cities $c_1, c_2, \ldots, c_n$
- distance between city i and j, d_{ij}

Find the shortest tour.



Assume a very fast PC:

1 flop = 1 nanosecond
= 10^{-9} sec.
= 1,000,000,000 ops/sec
= 1 GHz.

A tour requires n-1 additions. How many different tours?

Choose the first city n ways,
the second city n-1 ways,
the third city n-2 ways,
etc.

tours = n (n-1) (n-2) . . . .(2) (1) = n! (*Combinations*)

Total number of additions = n(n-1)! (*Rule of Product*)

If n=8, T(n) = 8•7! = 40,320 flops < 1/3 second.

HOWEVER . . . . . .

If n=50, T(n) = 50•49!
= $3.04\ 10^{64}$
= $3.04\ 10^{55}$ seconds
= $5.0\ 10^{53}$ minutes
= $8.0\ 10^{51}$ hours
= $3.0\ 10^{50}$ days
= $4.0\ 10^{49}$ weeks
= $7.0\ 10^{47}$ years.
...a long time. You'll be an old person (dead) before it's finished!

There are some problems for which we do not know if efficient algorithms exist to solve them!

Section 1.1: Logic

proposition : true = T (or 1) or false = F (or 0) (binary logic)

- 'the moon is made of green cheese'
- ' go to town!' X - imperative
- 'What time is it?' X – interrogative

propositional variables: P, Q, R, S, . . .

New Propositions from old: calculus of propositions - relate new propositions to old using TRUTH TABLES

- Logical operators: unary, binary
- Unary:
  - Negation
- Binary
  - Conjunction
  - Disjunction
  - Exclusive OR
  - Implication
  - Biconditional

- Unary
  - Negation

'not'
Symbol: ¬

Example: P: I am going to town
¬ P: I am not going to town;
It is not the case that I am going to town;
I ain't goin'.

Truth Table

P	¬ P
F(0)	T(1)
T(1)	F(0)

- Binary
  - Conjunction: 'and'
    Symbol: $\wedge$

Truth Table

P	Q	P $\wedge$ Q
0	0	0
0	1	0
1	0	0
1	1	1

Example: P - 'I am going to town'
Q - 'It is going to rain'

P $\wedge$ Q: 'I am going to town and it is going to rain.'

Note: Both P and Q must be true!!!!!

- Binary
  - Disjunction: *inclusive* 'or'
    Symbol: $\vee$

Example: P - 'I am going to town'
Q - 'It is going to rain'

P $\vee$ Q: 'I am going to town or it is going to rain.'

Note: Only one of P and Q must be true.
Hence, the *inclusive* nature.

Truth Table:

P	Q	P $\vee$ Q
0	0	0
0	1	1
1	0	1
1	1	1

- Binary
  - Exclusive OR: Symbol ⊕

Truth Table

P	Q	P ⊕ Q
0	0	0
0	1	1
1	0	1
1	1	0

Example:

P - 'I am going to town'

Q - 'It is going to rain'

P ⊕ Q: 'Either I am going to town or it is going to rain.'

Note: Only one of P and Q must be true.

- Binary
  - Implication: 'If...then...'
    Symbol: →

Truth Table

P	Q	P → Q
0	0	1
0	1	1
1	0	0
1	1	1

Example:

P - 'I am going to town'

Q - 'It is going to rain'

P → Q: 'If I am going to town then it is going to rain.'

- Implication (cont.)

  Equivalent forms:
  - If P, then Q
  - P implies Q
  - If P, Q
  - P only if Q
  - P is a sufficient condition for Q
  - Q if P
  - Q whenever P
  - Q is a necessary condition for P

  Note: The implication is false only when P is true and Q is false!

There is no causality implied here!

'If the moon is made of green cheese then I have more money than Bill Gates' (T)

'If the moon is made of green cheese then I'm on welfare' (T)

'If 1+1=3 then your grandma wears combat boots' (T)

'If I'm wealthy then the moon is not made of green cheese.' (T)

'If I'm not wealthy then the moon is not made of green cheese.' (T)

Terminology:

P = premise, hypothesis, antecedent
Q = conclusion, consequence

More terminology:

$Q \rightarrow P$ is the CONVERSE of $P \rightarrow Q$

$\neg Q \rightarrow \neg P$ is the CONTRAPOSITIVE of $P \rightarrow Q$

Example:

Find the converse and contrapositive of the following statement:

R: 'Raining tomorrow is a sufficient condition for my not going to town.'

Step 1: Assign propositional variables to component propositions
P: It will rain tomorrow
Q: I will not go to town

Step 2: Symbolize the assertion
R: P→Q

Step 3: Symbolize the converse
Q→P

Step 4: Convert the symbols back into words
'If I don't go to town then it will rain tomorrow'
or
'Raining tomorrow is a necessary condition for my not going to town.'
or
'My not going to town is a sufficient condition for it raining tomorrow.'

- Binary
  - Biconditional: 'if and only if', 'iff'
    Symbol: ↔

Truth Table

P	Q	P ↔ Q
0	0	1
0	1	0
1	0	0
1	1	1

Example: P - 'I am going to town', Q - 'It is going to rain'

P ↔ Q: 'I am going to town if and only if it is going to rain.'

Note: Both P and Q must have the same truth value.

- Imprecision of the natural language:
  'If you finish your meal then you can have dessert'

- Breaking assertions into component propositions - look for the logical operators!

  Example: 'If I go to Harry's or go to the country I will not go shopping.'

  P: I go to Harry's
  Q: I go to the country
  R: I will go shopping

  If.....P.....or.....Q.....then....not.....R

  $(P \vee Q) \rightarrow \neg R$

Constructing a truth table:

- one column for each propositional variable
- one for the compound proposition
- count in binary
- n propositional variables = 2^n rows

You may find it easier to include columns for propositions which themselves are component propositions.

Truth Table

P	Q	R	(P V Q)→¬R
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0