Chapter 1 Proposition Logic

Sorting and Time Complexity

Insertion sort

Look at first unsorted element (second card) and place in its correct location by pulling the other cards out and place the card in its correct spot. This can take a long time if you need to move many items.

For n cards, we apply this sort n times — once per card.

Later on the sort we have to move up to n cards to make space — n/2 on average

Quick sort (A Faster Method)

Choose the middle element and the place the others around to give the cards an *approximate* sorted order

For n cards, we apply the overall strategy about log2(n) **base 2** times

No possible "movement penalty" as the cards are only swapped. n/2 swaps are possible in each step

This method starts taking longer than an insertion sort, but it does not grow as fast.

1.1 Propositional Logic

Proposition: Statement that is either True or False. Very black and white

Proposition:

`It is raining. It is sunny. 1+1=2`

`X + 1 > 3 is *Not* a Proposition! We are missing some details`

p, q, r, s… are some variables used in Propositional logic

> p = `1 + 1 = 2`
> q = `It is sunny.`

r = p ˆ q: p `and` q. The ˆ is a conjunction

p	q	p^q
T	T	T
T	F	F
F	T	F
F	F	F

p v q: p `or` q

p	q	p v q
T	T	T
T	F	T
F	T	T
F	F	F

This `or` is also known as **inclusive** `or`.

There is an **exclusive** `or`.

t: -p. `Not` p. Symbol is actually a '-' with a little leg. Could be a ~ as well.

p	-p
T	F
F	T

The `not` is just the opposite of p

p: The Gators will not win the national title

q: The Gators will get a new head coach

-p^q: The Gators will not win the national title but they will get a new coach[1]

u: :p->q: if the Gators do not win the national title they will get a new head coach. the `->` means implies. `If p then q`, `If p, q`

p	q	p->q
T	T	T
T	F	F
F	T	T
F	F	T

Only way to logicalally disprove, must be p and not q. This is testing if p \`implies\` q

p<->q: If and only if. `Bidirectional` implication. Also known as `iff`

p	q	p<->q
T	T	T
T	F	F
F	T	F
F	F	T

If and only if p is true, q is true. If they are both true, there is nothing to disprove the assumption

r: p->q

s: q->q

t: -q->-p

p	q	r	s	t
T	T	T	T	T
T	F	F	T	F
F	T	*T*	*F*	T
F	F	T	T	T

`Logically falacy of `converse` [2]. r<->t`

p + q: : `+ with a circle around it` is exclusive or. `p <-> -q`

1.2 English to Propositional Logic

Example:

> You cannot ride the roller coaster if you are under 4 feet tall unless you are older than 16 years old

`q:  You can ride the roller coaster ride`

`r:  You are under 4 feel tall`

`s:  You are older than 16 yrs.`

(r ^ -s) -> -q

Example:

`q:  You can see the movie`

`r:  You are over 18 years old`

`s:  You have the permission of the parent`

r v q <-> q

The way the book words it...

q -> r v s

q	r	s
T	T	T
T	T	F
T	T	F
F	F	F

Search Engines

Search: *university gainesville -florida* to exclude `florida`

Search: *"university gainesville"* to search `only university gainesville`

Logic Puzzles

Example 7: Knights always tell the truth

`p: Knight`

`-p: Knave`

a:"B is a knight"

b:"The two of us are opposite types"

(a and -b) or (-a and b)

a	b	a's statement	b's statement	Possible?
T	T	T	F	No-b lied
T	F	F	T	No-a lied, b told the truth
F	T	T	T	No-a can't tell the truth
F	F	F	F	Yes

Both A and B are knaves

1.3 Propositional Equivalences

Tautology: `p or -p`. Always true

Contradiction: `p and -p`. Always False

Find a case where `p<-->q` is a tautology for some p and q. Also known as `p===q`, or they are logically equivalent to each other.

De Morgans Laws

`-(p or q) === -p and -q`. Distributing the '-'

`-(p and q) === -p or -q`. Distributing the '-'

`-(p v q v r)` === `-p ^ -q ^ -r`

Logical Statements

‘P and True’ is P

‘P or False’ is P # 1.4

`x = 1 + 4` itself is not a Proposition. It is, however, a function that translates into a proposition when we give it certain values.

`P(x,y)`

To make it a proposition we need to give the function values for x and y.

`P(1,2)` is False

`P(2,1)` is True

`x = 1 + 4` Is known as the `Predicate` for this example

Domain and Range

`f(x) = x^2`

Range is either True or False for predicate logic. Domain is what x represents, for example, a student

`Upside-down A:` “For all”

`Upside-down E:` “There exists at least one”

`Ev(xer) P(x,2)`. Upside-down e base for all real numbers. There is at least one number to make this statement true. If using the upside-down a, this would be false because not *every* number makes the statement P(x,2) true.

`Double-r` symbol represents all the reals

`E` symbol represents *within*

Example

`P(x):  x can speak Russian.`

`Q(x):  x knows c++`

There exists one student at UF who knows both.

1.5 Nested Quantifiers

Recall that upside down A means all and upside down E means there exists one

Sometimes you want to combine the E and A

`X is the students`

`Y is the course`

We can say: `Ex Ay Enrolled(x,y)` There exists some student enrolled in every course

We can say: `Ax Ey Enrolled(x,y)` Each student is enrolled in at least one course

We can say: `Ay Ex Enrolled(x,y)` Every course has at least one enrolled student

THE ORDER MATTERS

`not ExAy Enrolled(x,y)` is the same as `Ax not [Ay enrolled(x,y)]` which is the same as `Ax Ey not Enrolled(x,y)`

`x = students`

`y = class`

`C(Randy Goldbery, CS252)`, *Randy is enrolled in CS 252*

`Ex(X(x,Math 222) ^ C(x, CS 252))`, *There exists a student that is enrolled in both Math 222 and CS 252*

1.6 Rules of Inference

Using De Morgans laws we can make inferences about statements. We can then infer certain conclusions based on those inferences.

`p = You have a current password`

`L = You can log onto the network`

We can infer..

- `p -> L`
- `p`
- Therefore (three dots) `L` is true

Prove that it rained

What we know..

- `r = It rains.`
- `f = It is foggy.`
- `s = The sailing race will be held.`
- `d = The life-saving demo will go on.`
- `t = The trophy will be awarded.`
- `-r or -f -> s and d`
- `s -> t`
- `-t`

We can infer..

Since we know that the trophy was not awarded (`-t`), and `s -> t`, we can say `-s`.

We now know that the right hand side of `-r or -f -> s and d` must be false, therefore `-(-r or -f)` and De Morgan's laws tells us `(r and f)`

`-(r or -f)` can be written as `(r and f)` which implies `r` must be true. Therefore **it rained**.

1.7 Proofs

Proofs: A logical argument to prove a certain condition

axioms/postulates: Must not cause circular reasoning

Direct Proof

If n is odd, n^2 is odd

`n = 2k + 1` where k is an int

`n^2 = (2k +1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1`

`n^2 = 2c + 1` (c is a constant).

n^2 is odd

Proof by Contraposition

`p` therefore `p -> q`

`-q`

`-q -> -p` therefore, `p -> q`

Proof by Contradiction

If the right hand side is false, then the left hand side must also be false

1.8 Further proof techniques and approaches

`n is an int`

Show that n^2 >= n

n	result
+	+
0	0
-	-

For n is positive `n >= 1`, multiply both sides by `n` and `n^2 >= n`

For n is negative `n^2 !< 0` and `n <= 1` therefore `n <= n^2`

Without loss of generality: If we state something a bit more general than it actual is, it can still apply to sub cases of the same argument

There does exist an n such that…

`n s.t.  n^2 = 4`

This is known as proof by example or a *constructive* proof

Show that x and y are both irrational and exist s.t. x^y is rational.

`sqrt(2)^2 = 2`

`sqrt(2)^sqrt(2)^sqrt(2) = 2`

`sqrt(2)^sqrt(2)` is the x value and `^sqrt(2)` is the y. This assumption works proving that there is at least one case that exists which is enough for the proof.

`sqrt(2)^sqrt(2) = ?`, this is the x value

Since we cannot show the exact x to use, this is known as a *nonconstructive* proof.

`oooooo`

`oooooo`

`ooooo.`

Whoever takes the `.` loses the game. You just take 1-3 stones. You cannot say for certain that you or your opponent can win. If all that remains on the board is the `.`, the next player to go for sure loses.

Chapter 2

2.1 Sets

Sets vs Lists

Set	List
Unordered	Ordered
No duplicates	Duplicates Allowed

Set Examples

- `S = {0,1,2,3,4,5}`
- `T = {x: x>0}`
- `N` (fancy N) refers to the natural numbers '{0,1,2,3,4}
- *`Z` (fancy Z) refers to all numbers {-2,-1,0,1,2}. You can specify `Z+` to only refer to the positive.*
- *`R` (fancy R) refers to all the real numbers `{x: x is a real number}`*

- `Q` (fancy Q) refers to all rationals
- `C` (fancy C) refers to all complex numbers