Mathematical Logic 2016

Lecture 2: Propositional logic - Syntax and Semantics

Instructor: Ashutosh Gupta

TIFR, India

Compile date: 2016-08-03

Propositional logic

Propositional logic

- deals with propositions,
- only infers from the structure over propositions, and
- does not look inside propositions.

Example 2.1

If the seed catalog is correct then if seeds are planted in April then the flowers bloom in July. The flowers do not bloom in July. Therefore, if seeds are planted in April then the seed catalog is not correct.

The above argument contains the following propositions

- c = *the seed catalogue is correct*
- s = *seeds are planted in April*
- f = *the flowers bloom in July*

If c *then if* s *then* f*. not* f*. Therefore, if* s *then not* c*.*

$$((c \Rightarrow (s \Rightarrow f)) \wedge \neg f) \Rightarrow (s \Rightarrow \neg c)$$

Analysis of such strings is propositional logic

Topic 2.1

Syntax

Propositional variables

We assume that there is a set **Vars** of countably many propositional variables.

- Since **Vars** is countable, we assume that variables are indexed.

$$\textbf{Vars} = \{p_1, p_2, \dots\}$$

- The variables are just names/symbols without inherent meaning
- We may also use $p, q, r, .., x, y, z$ to denote the propositional variables

Logical connectives

The following 10 symbols that are called logical connectives.

formal name	symbol	read as	
true	$\top$	top	0-ary symbols
false	$\bot$	bot	
negation	$\neg$	not	unary symbols
conjunction	$\wedge$	and	binary symbols
disjunction	$\vee$	or	
implication	$\Rightarrow$	implies	
equivalence	$\Leftrightarrow$	iff	
exclusive or	$\oplus$	xor	
open parenthesis	(		
close parenthesis	)		

We assume that the logical connectives are not in **Vars**.

Propositional formulas

A propositional formula is a finite string containing symbols in **Vars** and logical connectives.

Definition 2.1

The set of propositional formulas is the smallest set $\mathbf{P}$ *such that*

- $\top, \bot \in \mathbf{P}$
- *if* $p \in \mathbf{Vars}$ *then* $p \in \mathbf{P}$
- *if* $F \in \mathbf{P}$ *then* $\neg F \in \mathbf{P}$
- *if* $\circ$ *is a binary symbol and* $F, G \in \mathbf{P}$ *then* $(F \circ G) \in \mathbf{P}$

Definition 2.2 (Alternate presentation of the above definition)

$F \in \mathbf{P}$ *if*

$$F \triangleq p \mid \top \mid \bot \mid \neg F \mid (F \vee F) \mid (F \wedge F) \mid (F \Rightarrow F) \mid (F \Leftrightarrow F) \mid (F \oplus F)$$

where $p \in \mathbf{Vars}$.

Some notation

Definition 2.3

$\top, \bot$, *and* $p \in$ **Vars** *are atomic formulas.*

Definition 2.4

For each $F \in$ **P**, *let* **Vars**(F) *be the set of variables appearing in* F.

Exercise 2.1

"appear in" is not defined yet. Give a formal definition of the phrase.

Examples of propositional formulas

Exercise 2.2

Is the following belongs to **P***?*

- $\top \Rightarrow \bot$ ✗
- $(\top \Rightarrow \bot)$ ✓
- $(p_1 \Rightarrow \neg p_2)$ ✓
- (p_1)✗
- $\neg\neg\neg\neg\neg\neg\neg p_1$ ✓

Not all strings over **Vars** and logical connectives are in **P**.

How can we argue that a string does or does not belong to **P**?

We need a method to recognize a string belongs to **P** or not.

Parse tree

By def. 2.1, $F \in \mathbf{P}$ iff F is obtained by unfolding of the generation rules

Definition 2.5

A parse tree of a formula $F \in \mathbf{P}$ is a tree such that

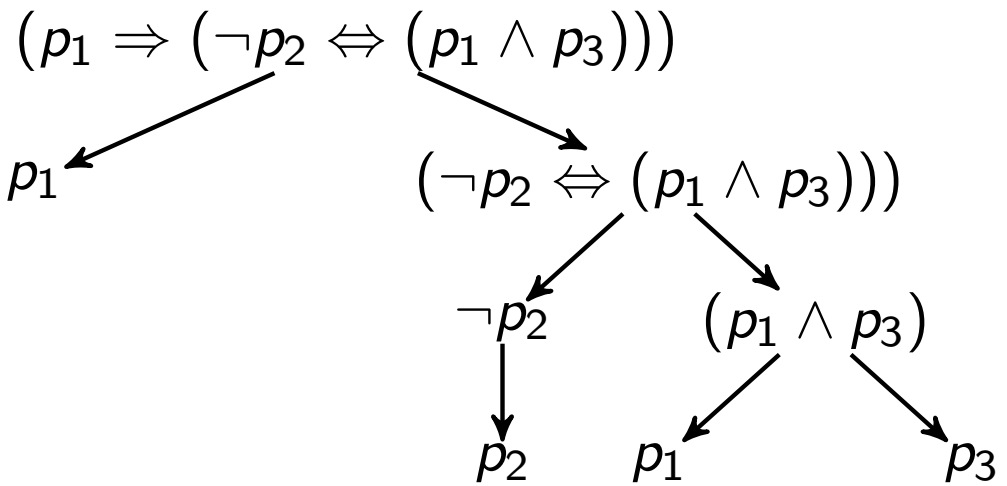
- *the root is F,*
- *leaves are atomic formulas, and*
- *each internal node is formed by applying some formation rule on its children.*

reverse direction is immediate. for forward direction, we will prove stronger theorem, i.e., existance of unique parsing tree

Theorem 2.1

$F \in \mathbf{P}$ iff there is a parse tree of F

Example 2.2



Principle of structural induction

In order to prove, such theorems we need to get used to the principle of structural induction.

Theorem 2.2

Every formula in $\mathbf{P}$ *has a property* Q *if*

- *Base case: every atomic formula has property* Q
- *induction steps: if* $F, G \in \mathbf{P}$ *have property* Q *so do* $\neg F$ *and* $(F \circ G)$*, where* $\circ$ *is a binary symbol*

Exercise 2.3

Prove theorem 2.2.

Topic 2.2

Unique parsing

Matching parenthesis

Theorem 2.3

Every $F \in \mathbf{P}$ has matching parenthesis, i.e., equal number of '(' and ')'.

Proof.

base case:
atomic formulas have no parenthesis. Therefore, matching parenthesis

induction steps:
We assume $F, G \in \mathbf{P}$ has matching parenthesis.
Let n_F and n_G be the number of '(' in F and G respectively.
Trivially, $\neg F$ has matching parenthesis.
For some binary symbol $\circ$, the number of both '(' and ')' in $(F \circ G)$ is $n_F + n_G + 1$.

Due to the structural induction, the property holds. □

Prefix of a formula

Theorem 2.4

A proper prefix of a formula is not a formula.

Proof.

We show a proper prefix of a formula is in one of the following forms.

1. strictly more '(' than ')',
2. a (possibly empty) sequence of $\neg$.

None of the above are clearly in **P**.

base case:

A proper prefix of atomic formulas is empty string, which is the second case

...

Exercise 2.4

Give examples of the above two cases

Prefix of a formula

Proof.

induction step:

Let $F, G \in \mathbf{P}$.

Consider proper prefix F' of $\neg F$. There are two cases.

- $F' = \epsilon$, case 2
- $F' = \neg F''$, where F'' is a proper prefix of F. Now we again have two subcases for F''.
  - If F'' is in case 1 then F' belongs to case 1
  - If $F'' = \neg..\neg$ then F' belongs to case 2

Consider proper prefix F' of $(F \circ G)$. There are several cases.... □

Exercise 2.5

Complete the above proof.

Unique parsing

Theorem 2.5

Each $F \in \mathbf{P}$ has a unique parsing tree.

Proof.

$\nu(F) \triangleq$ number of logical connectives in F. We apply induction over $\nu(F)$.
base case: $\nu(F) = 0$
F is an atomic formula, therefore has a single node parsing tree.
inductive steps: $\nu(F) = n$
We assume that each F' with $\nu(F') < n$ has a unique parsing tree.
case $F = \neg G$: Since G has a unique parsing tree, F has a unique parsing tree.
case $F = (G \circ H)$:
Suppose there is another formation rule s.t. $F = (G' \circ' H')$.
Since $F = (G \circ H) = (G' \circ' H'),\ G \circ H) = G' \circ' H')$.
Wlog, G is prefix of G'.
Since $G, G' \in \mathbf{P}$, G can not be proper prefix of G'. Therefore, $G = G'$.
Therefore, $\circ = \circ'$. Therefore, $H = H'$. Therefore, one way to unfold F.
F has a unique parsing tree. □

Parsing algorithm

The previous proofs suggest a parsing algorithm to generate parsing tree.

Algorithm 2.1: PARSER

Input: F : a string over **Vars** and logical connectives
Output: parse tree if $F \in \mathbf{P}$, exception FAIL otherwise
if $F = p$ *or* $F = \top$ *or* $F = \bot$ **then** **return** $(\{F\}, \emptyset)$;
if $F = \neg G$ **then**
    $(V, E) := \text{PARSER}(G)$;
    return $(V \cup \{F\}, E \cup \{(F, G)\})$;
if $F = (F')$ **then**
    $G :=$ smallest prefix of F' where parenthesis match or atomic formula after a sequence of negation symbols;
    $o'H := tail(F', len(G))$;
    $(V_1, E_1) := \text{PARSER}(G)$;
    $(V_2, E_2) := \text{PARSER}(H)$;
    return $(V_1 \cup V_2 \cup \{F\}, E_1 \cup E_2 \cup \{(F, G), (F, H)\})$;
else
    Throw FAIL

Parse tree is a DAG

We have been thinking that the parsing algorithm produces parse tree.

However, the previous algorithm produced a parse DAG, because it maintains a set of formulas as node of the parse tree.

Subformula

Definition 2.6

A formula G is a subformula of formula F if G occurs within F. G is a proper subformula of F if $G \neq F$. Let $sub(F)$ denote the set of subformulas of F.

All the nodes of the parse tree of F is the set of subformulas of F.

Definition 2.7

Immediate subformulas are the children of a formula in its parse tree.
And, leading connective is the connective that is used to join the children to the formula.

Example 2.3

Consider $F = (p_1 \Rightarrow (\neg p_2 \Leftrightarrow (p_1 \wedge p_3)))$

$sub(F) = \{(p_1 \Rightarrow (\neg p_2 \Leftrightarrow (p_1 \wedge p_3))), (\neg p_2 \Leftrightarrow (p_1 \wedge p_3)), \neg p_2, (p_1 \wedge p_3),$
$p_1, p_2, p_3\}$

The leading connective of F is $\Rightarrow$.

Topic 2.3

Shorthands

Too many parenthesis

In the above syntax, we need to write a large number of parenthesis.

Using precedence order over logical connectives, we may drop some parenthesis without loosing the unique parsing property.

Example 2.4

Consider $((p \wedge q) \Rightarrow (r \vee p))$

- *We may drop outermost parenthesis without any confusion*

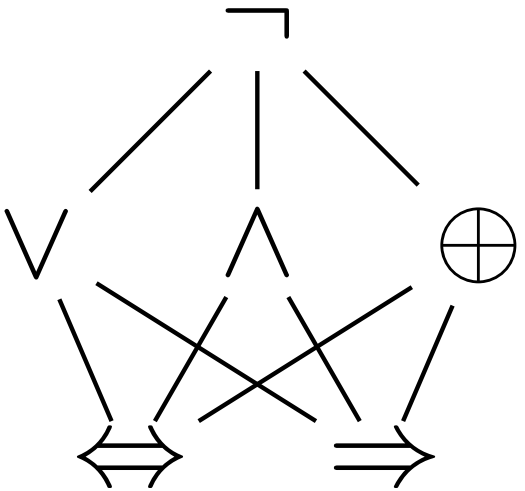
$$(p \wedge q) \Rightarrow (r \wedge p)$$

- *If* $\wedge$ *and* $\vee$ *get precedence over* $\Rightarrow$ *in unfolding during parsing then we do not need the rest of parenthesis*

$$p \wedge q \Rightarrow r \wedge p$$

Precedence order

We will use the following precedence order in writing the propositional formulas



Using precedence order

Consider the following formula for $n > 1$

$$F_0 \circ_1 F_1 \circ_2 F_2 \cdots \circ_n F_n,$$

where F_0,..,F_n are either atomic or enclosed by parenthesis, or their negation.

We transform the formula as follows

- Find a o_i such that o_{i-1} and o_{i+1} have lower precedence.
- Introduce parenthesis around $(F_{i-1} \circ_i F_i)$ and call it F'_i.

$$F_0 \circ_1 \ldots F_{i-2} \circ_{i-1} F'_i \circ_{i+1} F_{i+1} \cdots \circ_n F_n$$

We apply the above until $n = 1$ and then apply the normal parsing procedure.

Inside of F_is may also have similar ambiguities, which are recursively resolved using the above procedure.

Exercise 2.6

Write a formal procedure from the informal description above.

Example precedence order

Example 2.5

Which of the following formulas can be unambiguously parsed?

- $\neg p \vee (p \oplus q) \Leftrightarrow p \wedge q$ ✓
- $p \vee q \wedge r$ ✗
- $p \vee q \vee r$ ✗
- $p \Rightarrow q \Rightarrow r$ ✗

Associativity preference may further reduce the need of parenthesis

Exercise 2.7

Modify the parsing procedure of the previous slide to incorporate associativity preference.

Substitution

Definition 2.8

For $F \in \mathbf{P}$ and $p_1, \ldots, p_k \in \mathbf{Vars}$, let $F[G_1/p_1, \ldots, G_k/p_k]$ denote another formula obtained by simultaneously replacing all occurrence of p_i by a formula G_i for each $i \in 1..k$.

For short hand, we may write a formula as $F(p_1, \ldots, p_k)$, where we say that $p_1, \ldots, p_k$ are the variables that play a special role in the formula F. Let $F(G_1, \ldots, G_n)$ be $F[G_1/p_1, \ldots, G_k/p_k]$.

Example 2.6

1. $(p \Rightarrow (r \Rightarrow p))[(r \oplus s)/p] = ((r \oplus s) \Rightarrow (r \Rightarrow (r \oplus s)))$
2. $(p \Rightarrow (r \Rightarrow p))[(r \oplus s)/p, x/r] \neq (p \Rightarrow (r \Rightarrow p))[(r \oplus s)/p][x/r]$

Exercise 2.8

a. The definition 2.8 is informal. Give a formal definition.

b. Write the result of substitutions in the second example.

Topic 2.4

Semantics

Truth values

We denote the set of truth values as $\mathcal{B} \triangleq \{0, 1\}$.

0 and 1 are only distinct objects without any intuitive meaning.

We may view 0 as false and 1 as true but this is only our emotional response to the symbols.

Model

Definition 2.9

A model is an element of $\mathbf{Vars} \to \mathcal{B}$.

Since **Vars** is countable, the set of models is non-empty, and infinitely many.

A model m may or may not satisfy a formula F.
The satisfaction relation is usually denoted by $m \models F$ in infix notation.

Propositional Logic Semantics

Why is "smallest relation" not mentioned?

Definition 2.10

The satisfaction relation $\models$ between models and formulas is the relation that satisfies the following conditions.

$$
\begin{aligned}
m &\models \top \\
m &\models p && \text{if } m(p) = 1 \\
m &\models \neg F && \text{if } m \not\models F \\
m &\models F_1 \vee F_2 && \text{if } m \models F_1 \text{ or } m \models F_2 \\
m &\models F_1 \wedge F_2 && \text{if } m \models F_1 \text{ and } m \models F_2 \\
m &\models F_1 \oplus F_2 && \text{if } m \models F_1 \text{ or } m \models F_2, \text{ but not both} \\
m &\models F_1 \Rightarrow F_2 && \text{if if } m \models F_1 \text{ then } m \models F_2 \\
m &\models F_1 \Leftrightarrow F_2 && \text{if } m \models F_1 \text{ iff } m \models F_2
\end{aligned}
$$

Theorem 2.6

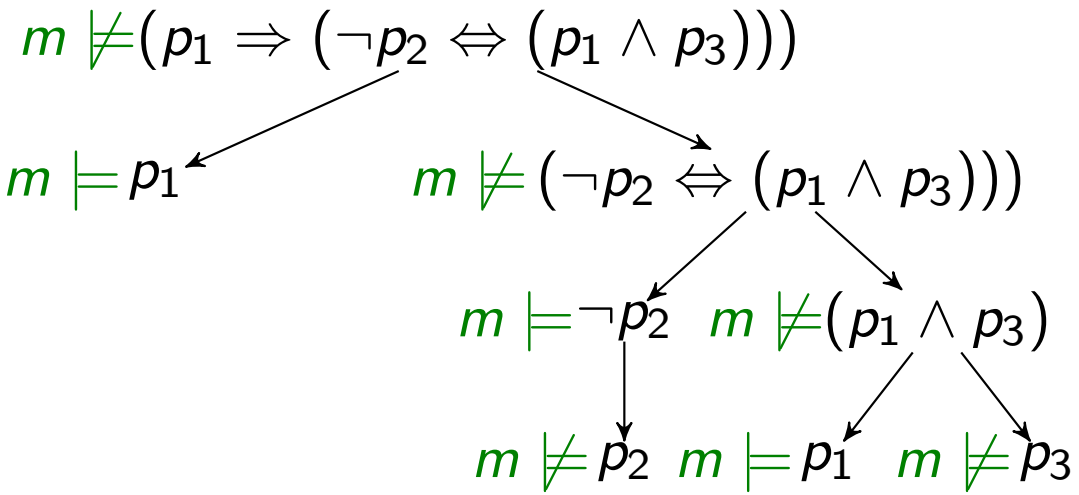
There is exactly one relation that satisfies the above conditions.

Proof. Since each $F \in \mathbf{P}$ has a unique parse tree, we have a terminating procedure to check if $m \models F$ or not.

Example: satisfaction relation

Example 2.7

Consider model $m = \{p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 0, \dots\}$
And, formula $(p_1 \Rightarrow (\neg p_2 \Leftrightarrow (p_1 \wedge p_3)))$



Exercise 2.9

write the satisfiability checking procedure formally.

Satisfiable, valid, unsatisfiable

We say

- m *satisfies* F if $m \models F$,
- F is *satisfiable* if there is a model m s.t. $m \models F$,
- F is *valid* (written $\models F$) if for each model m $m \models F$, and
- F is *unsatisfiable* (written $\not\models F$) if there is no model m s.t. $m \models F$.

Exercise 2.10

If F is sat then $\neg F$ is _____.
If F is valid then $\neg F$ is _____.
If F is unsat then $\neg F$ is _____.

Implication

We extend the usage of $\models$.

Definition 2.11

Let M be a (possibly infinite) set of models.
$M \models F$ if for each $m \in M$, $m \models F$.

Definition 2.12

Let Σ be a (possibly infinite) set of formulas.
$\Sigma \models F$ if for each model m that satisfies each formula in Σ, $m \models F$.

$\Sigma \models F$ is read Σ implies F.
If $\{G\} \models F$ then we may write $G \models F$.

Theorem 2.7

$F \models G$ iff $\models (F \Rightarrow G)$.

Equisatisfiable

Definition 2.13

Let $F \equiv G$ *if* $m \models F$ *iff* $m \models G$.

Theorem 2.8

$F \equiv G$ *iff* $\models (F \Leftrightarrow G)$.

Topic 2.5

Decidability of SAT

Partial models

Let $m|_{\mathbf{Vars}(F)} : \mathbf{Vars}(F) \to \mathcal{B}$ and for each $p \in \mathbf{Vars}(F)$, $m|_{\mathbf{Vars}(F)}(p) = m(p)$

Theorem 2.9

If $m|_{\mathbf{Vars}(F)} = m'|_{\mathbf{Vars}(F)}$ *then* $m \models F$ *iff* $m' \models F$

Proof sketch.

the procedure to check $m \models F$ only looks at the $\mathbf{Vars}(F)$ part of m. Therefore any extension of $m|_{\mathbf{Vars}(F)}$ will have same result as $m \models F$ or $m \not\models F$. □

Definition 2.14

We will call elements of $\mathbf{Vars} \hookrightarrow \mathcal{B}$ *as partial models.*

Exercise 2.11

Write the above proof formally.

Propositional satisfiability problem

The following problem is called the satisfiability problem

For a given $F \in \mathbf{P}$, is F satisfiable?

Theorem 2.10

The propositional satisfiability problem is decidable.

Proof.

Let $n = |\mathbf{Vars}(F)|$.
We need to enumerate 2^n elements of $\mathbf{Vars}(F) \to \mathcal{B}$.
If any of the models satisfy the formula, then we can always extend to a the full model and F is sat
Otherwise, F is unsat. □

Exercise 2.12

Give a procedure to decide the validity of a formula.

Topic 2.6

Truth tables

Truth tables

Truth tables was the first method to decide propositional logic.

The method is usually presented in slightly different notation.

We need to assign a truth value to every formula.

Truth function

A model m is in $\mathbf{Vars} \to \mathcal{B}$.

We can extend m to $\mathbf{P} \to \mathcal{B}$ in the following way.

$$m(F) = \begin{cases} 1 & m \models F \\ 0 & otherwise. \end{cases}$$

The extended m is called truth function.

Since truth functions are natural extensions of models, we did not introduce new symbols.

Truth functions for logical connectives

Let F and G are logical formulas, and m is a model.
Due to the semantics of the propositional logic, the following holds for the truth functions.

$m(F)$	$m(\neg F)$
0	1
1	0

$m(F)$	$m(G)$	$m(F \wedge G)$	$m(F \vee G)$	$m(F \oplus G)$	$m(F \Rightarrow G)$	$m(F \Leftrightarrow G$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Exercise 2.13

Verify the above table against the definition of $\models$

Truth table

For a formula F, a truth table consists of $2^{|\mathbf{Vars}(F)|}$ rows. Each row considers one of the partial models and computes the truth value of F for each model.

Example 2.8

Consider $(p_1 \Rightarrow (\neg p_2 \Leftrightarrow (p_1 \wedge p_3)))$

We will not write $m(.)$ *in the top row for brevity.*

p_1	p_2	p_3	$(p_1$	$\Rightarrow$	$(\ \neg$	p_2	$\Leftrightarrow ($	p_1	$\wedge$	$p_3\ ))$
0	0	0	0	1	1	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	1
0	1	0	0	1	0	1	1	0	0	0
0	1	1	0	1	0	1	1	0	0	1
1	0	0	1	0	1	0	0	1	0	0
1	0	1	1	1	1	0	1	1	1	1
1	1	0	1	1	0	1	1	1	0	0
1	1	1	1	0	0	1	0	1	1	1

The column under the leading connective has 1s therefore the formula is sat.

But, there are some 0s in the column therefore the formula is not valid.

Tedious truth tables

- We need to write 2^n rows even if some simple observations about the formula may prove unsatisfiablity/satisfiability.
  For example,
  - $(a \vee (c \wedge a))$ is sat (why? - no negation)
  - $(a \vee (c \wedge a)) \wedge \neg(a \vee (c \wedge a))$ is unsat (why?- contradiction at top level)
- We should be able to take such shortcuts?

We will see many methods that will allow us to take such shortcuts. But not now!

Topic 2.7

Problems

Parse Algorithm

Exercise 2.14

Show the run of Algorithm 2.1 on the following formulas.

1. $\neg q \Rightarrow (p \oplus r \Leftrightarrow s)$
2. $(\neg(p \Rightarrow q) \wedge (r \Rightarrow (p \Rightarrow q)))$

Let expression

We may extend the grammar of proportional logic with let expressions.

$$F := \cdots \mid (\text{let } p = F \text{ in } F)$$

Let-expression is a syntactic device to represent large formulas succinctly.

$$(\text{let } p = F \text{ in } G) \text{ represents } G[F/p]$$

Example 2.9

$$(let\ p = (q \wedge r)\ in\ ((p \wedge s) \vee (q \Rightarrow \neg p)))$$
$$represents$$
$$((q \wedge r) \wedge s) \vee (q \Rightarrow \neg(q \wedge r))$$

Exercise 2.15

Give an example in which let expressions allow us to represent a formula in exponentially less space.

Precedence order

Exercise 2.16

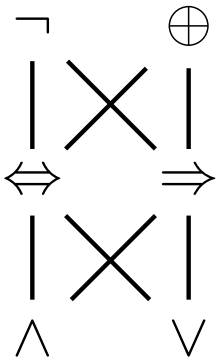
Add minimum parenthesis in the following formulas such that it has unique parsing under our precedence order

1. $p \wedge q \vee r \wedge s \wedge t \vee u \vee v \wedge w$
2. $p \Rightarrow \neg q \oplus p \vee p \wedge \neg r \Leftrightarrow s \wedge t$

Custom Precedence order

Exercise 2.17

Consider the following precedence order



Add minimal parentheses in the following formulas such that they have unique parsing tree

1. $\neg p \Rightarrow q \wedge r \Rightarrow p \Rightarrow q$
2. $p \Rightarrow \neg q \oplus p \vee p \wedge \neg r \Leftrightarrow s \wedge t$

Semantics

Exercise 2.18

Show $F(\bot/p) \wedge F(\top/p) \models F \models F(\bot/p) \vee F(\top/p)$.

Truth tables

Exercise 2.19

Prove/disprove validity of the following formulas using truth tables.

1. $(p \vee q) \Leftrightarrow \neg(\neg p \wedge \neg q)$
2. $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
3. $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (q \wedge r)$
4. $(p \Rightarrow (q \Rightarrow r)) \Leftrightarrow ((p \wedge q) \Rightarrow r))$
5. $p \wedge (q \oplus r) \Leftrightarrow (p \wedge q) \oplus (q \wedge r)$
6. $\bot \Rightarrow F$ *for any* F

End of Lecture 2