CS228 Logic for Computer Science 2020

Lecture 2: Propositional logic - syntax and parsing

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-01-17

Topic 2.1

Propositional logic - Syntax

Syntax

We need a quick method of identifying if a group of symbols is a logical argument.

We usually define a syntax.

Example 2.1

Grammar of English

Let us define syntax for propositional logic

Propositions

The logic is over a list of propositions.

- Sky is blue
- Sun is hot
- ... *many more*

We do not care what each one says. We give each one of them a symbol.

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
- Propositional variables are also called Boolean variables

Commentary: All results presented in this course are extendable to uncountable **Vars**. For the uncountable setting, we need transfinite induction. We will ignore those extensions.

Logic connects the variables

A logical argument connects the propositions.

Let us list all the possible ways of connecting them.

True and false

We should be able to talk about

- always true statement
- always false statement

Example 2.2

- *An apple is an apple* — *always true*
- *I like Apple and I do not like Apple* — *always false*

Logical connectives: Not, And, and Or

We may also need ability to say

- a statement that says negation of another
- two statements are true at the same time
- at least one of the two statements are true

Example 2.3

- *The apple is not sweet.*
- *The apple is sweet and Delhi is far.*
- *The apple is sweet or Delhi is far.*

More logical connectives: Implies, equality, and disequality

We may also need ability to say

- Implication
  if a statement is true then some other statement is also true
- Equivalence
  truth value of two statements are same
- Disequality
  truth value of two statements are different
  - Usually called exclusive or, meaning exactly one of the two is true

Example 2.4

- *If I work then I make money.* *(implication)*
- *I like an apple if and only if I like a pen.* *(equivalence)*
- *A is here or B is here, but both are not here.* *(exclusive or)*

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	(		punctuation
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* $(F \circ G) \in \mathbf{P}$

Some notation

Definition 2.2

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

Definition 2.3

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

Examples of propositional formulas

Exercise 2.1

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.

Example : symbolic argument

Example 2.5

We have seen the following argument.

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

where

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

We can write the above argument as propositional formula as follows

$$\Big(\Big(\underbrace{(c \Rightarrow (s \Rightarrow f))}_{\text{Premise 1}} \wedge \underbrace{\neg f}_{\text{Premise 2}} \Big) \Rightarrow \underbrace{(s \Rightarrow \neg c)}_{\text{Conclusion}} \Big)$$

Example: symbolizing bad and good puzzle

Problem Context

Example 2.6

The good people always tell the truth and the not good people always tell a lie. Now let us consider the following puzzle.

There are two people A and B. A says, "I am not good or B is good". What are A and B?

Let us give symbols to propositions:

- $p_A = A$ *is good.*
- $p_B = B$ *is good.*

Therefore, we encode the puzzle as follows.

$$\underbrace{(\neg p_A \vee p_B)}_{\text{Statement of } A} \Leftrightarrow p_A$$

To solve the puzzle, we need a satisfying assignment to the formula.

Topic 2.2

Parsing formulas

Parse tree

$F \in \mathbf{P}$ iff F is obtained by unfolding of the generation rules

Definition 2.4

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

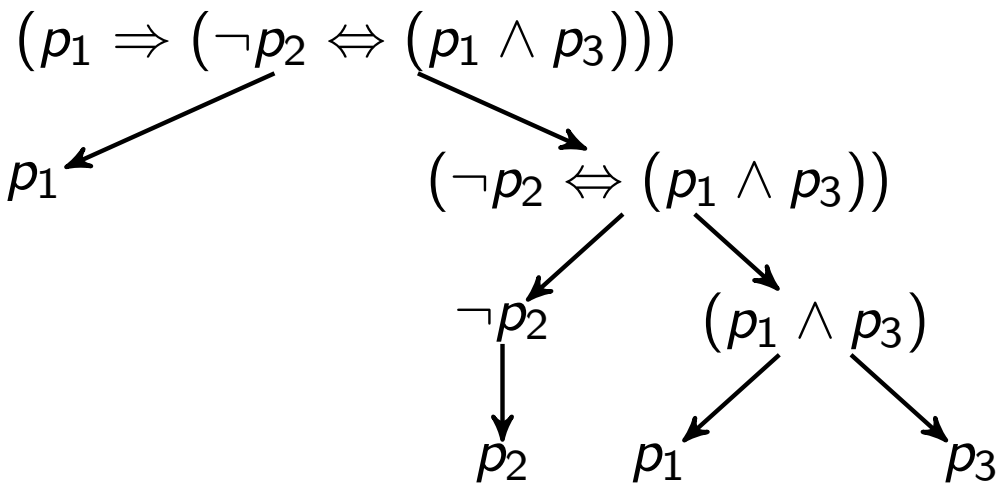
- *the root is F,*
- *leaves are atomic formulas, and*
- *each internal node is formed by applying some formation rule on its children.*

reverse direction is immediate. In forward direction, we can prove a 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 2.7



Unique parsing

Theorem 2.2

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

Proof.

We will post the proof online. □

Commentary: The proof is available for a curious mind and very much needed for the rigor of logic. However, we skipped the proof in the class.

Parse tree is a directed-acyclic graph (DAG)

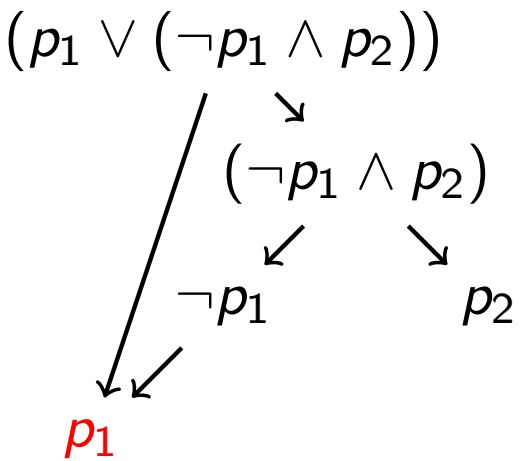
We have been thinking that the parsing produces parse tree.

However, the parsing produces a parse DAG.

Example 2.8

Consider formula $(p_1 \vee (\neg p_1 \wedge p_2))$

The following is the parse tree of the above formula.



Subformula

Definition 2.5

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.

The nodes of the parse tree of F form the set of subformulas of F.

Definition 2.6

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.

Example 2.9

Consider $F = (\neg p_2 \Leftrightarrow (p_1 \wedge p_3))$
$sub(F) = \{(\neg p_2 \Leftrightarrow (p_1 \wedge p_3)), \neg p_2, (p_1 \wedge p_3), p_1, p_2, p_3\}$
The leading connective of F is $\Leftrightarrow$.

Commentary: *Note that the above definition does not allow $p_2 \Leftrightarrow (p_1 \wedge p_3)$ to be a subformula of F, because $p_2 \Leftrightarrow (p_1 \wedge p_3)$ is not a formula. In later discussions, we may drop parenthesis in our writings and it may cause confusion. So, when we apply the above definition we need to keep the invisible parentheses in our mind.*

Topic 2.3

Shorthands

Too many parenthesis

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

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

Example 2.10

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

- *We may drop outermost parenthesis without any confusion*

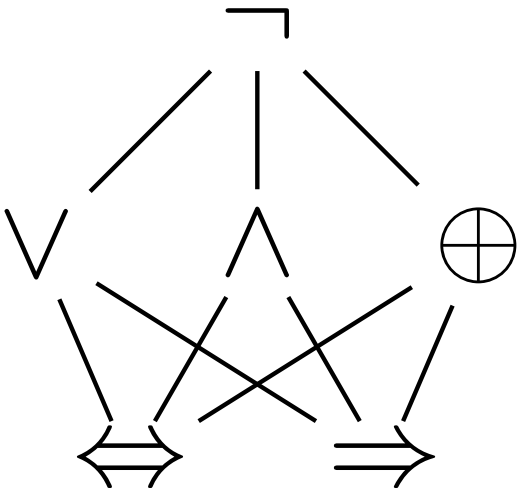
$$(p \wedge q) \Rightarrow (r \wedge p)$$

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

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

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 parentheses, or their negation.

We transform the formula as follows

- Find an o_i such that o_{i-1} and o_{i+1} have lower precedence if they exist.
- Introduce parentheses around $F_{i-1} \circ_i F_i$ and call it $F'_i = (F_{i-1} \circ_i F_i)$.

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

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

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

Example: Parsing using the precedence order

Example 2.11

Consider formula $p \wedge q \Rightarrow r \vee p$. *Let us try to bring back the parentheses.*

$\Rightarrow$ *has lower precedence then* $\wedge$, *therefore we can group neighbours of* $\wedge$

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

Since $\vee$ *has higher precedence over* $\Rightarrow$, *we first group* $\vee$.

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

Now we can group $\Rightarrow$ *without any confusion*

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

Example precedence order

Example 2.12

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

Associative

If a binary operator repeats, we do not know how to group.

We may give preference to one side or another. Let us make all our operators "right associative", i.e., first group the right occurrence.

Example 2.13

Consider formula $p \Rightarrow q \Rightarrow r$.

We first group the right $\Rightarrow$.

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

Then, we group the left $\Rightarrow$.

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

Exercise 2.2

Modify the parsing procedure of the earlier slide to support the above.

Substitution

Definition 2.7

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 $i \in 1..k$.

Example 2.14

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 2.3

a. The definition 2.7 is informal. Give a formal definition.

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

Notation for substitution

For shorthand, we may write a formula F as

$$F(p_1, \ldots, p_k),$$

where we say that variables $p_1, \ldots, p_k$ play a special role in F.

$$\text{Let } F(G_1, \ldots, G_n) \text{ be } F[G_1/p_1, \ldots, G_k/p_k].$$

Example 2.15

Let $F(p, q) = \neg p \oplus q$

$F(r \vee q, \top) = \neg(r \vee q) \oplus \top$

Commentary: This notation is very useful in the case when we do not know F but want to talk about the substitutions in a convenient way.

Topic 2.4

Problems

Exercise: symbolizing bad and good puzzle

Exercise 2.4

People are either good or bad. The good people always tell the truth and the bad people always tell a lie. Now let us consider the following puzzle.

There are two people A and B. A said some thing, but we could not hear. B said, "A is saying that A is bad". What are A and B?

Encode the above puzzle into a propositional logic formula.

Let expression

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

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

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

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

Example 2.16

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

Exercise 2.5

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

Precedence order

Exercise 2.6

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

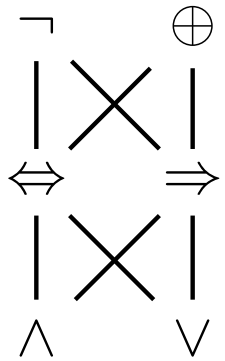
1. $p \wedge q \vee r \wedge s \wedge t \vee u \vee v \wedge w$
2. $p \Rightarrow \neg q \oplus p \vee p \wedge \neg r \Leftrightarrow s \wedge t$

Commentary: Please work the above problems with and without associative preference rules. In the exams, we will make it clear.

Custom precedence order

Exercise 2.7

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

Commentary: *Please work the above problems with and without associative preference rules. In the exams, we will make it clear.*

End of Lecture 2