Propositional Logic

Logan Borghi

University of Connecticut

May 13, 2022

Two Arenas of Logic

There are two separate focuses within the study of logic: the logic of formal languages, and the logic of natural languages. In this presentation we will focus on the logic of formal languages, and in particular we examine the language of propositional logic.

The logic of formal languages, often called mathematical logic is a discipline of mathematics, while the logic of natural languages instead lies closer to the domain of philosophy and linguistics.

An Outline

We begin by qualifying what constitutes letters and words in the language of propositional logic, and then proceed to define a few important related operations. With this we are able to express what a truth assignment is for a propositional sentence, and using truth assignments we define the truth table for a propositional sentence. Finally we see how a more mathematical treatment of logic allows for some nice formulas for truth tables of generalizations of the disjunction and conjunction.

The Language of Propositional Calculus

The Language of Propositional Calculus includes a countable set of propositional variables, sometimes called sentence letters, a collection of logical connectives, as well as left and right parentheses for readability.

$$\mathfrak{L}_{PC} := \{A_i | i \in \mathbb{N}\} \cup \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\} \cup \{(,)\}$$

Finite Sequences in $\mathfrak{L}_{PC}$

We have decided on a set of letters for the language of propositional calculus. But now we must decide on which words to allow as grammatical. Propositional sentences will be certain finite sequences with letters taken from $\mathfrak{L}_{PC}$... but which? First a definition.

$$x \text{ is a finite sequence on } \mathfrak{L}_{PC} :\Longleftrightarrow \exists m \in \mathbb{N} \quad x : m \to \mathfrak{L}_{PC}$$

And we write $x = \langle x(0), ..., x(m-1)\rangle = \langle x_0, ..., x_{m-1}\rangle$

Propositional Sentences in $\mathfrak{L}_{PC}$

Now we can define propositional sentences: intuitively ϕ is a propositional sentence if there is a way to "build it" from propositional variables using the logical connectives!
ϕ is a propositional sentence over $\mathfrak{L}_{PC} \iff \exists f = \langle \phi_0, ...\phi_m \rangle$ such that $\phi_m = \phi$ and for all $i \leq m$ either

- $\phi_i = \langle A_j \rangle, \quad j \in \mathbb{N}$
- $\phi_i = \phi_j \wedge \phi_k, \quad j, k < i$
- $\phi_i = \phi_j \vee \phi_k, \quad j, k < i$
- $\phi_i = \phi_j \rightarrow \phi_k, \quad j, k < i$
- $\phi_i = \phi_j \leftrightarrow \phi_k, \quad j, k < i$
- $\phi_i = \neg\phi_j, \quad j < i$
- $\phi_i = (\phi_j), \quad j < i$

An example of a propositional sentence

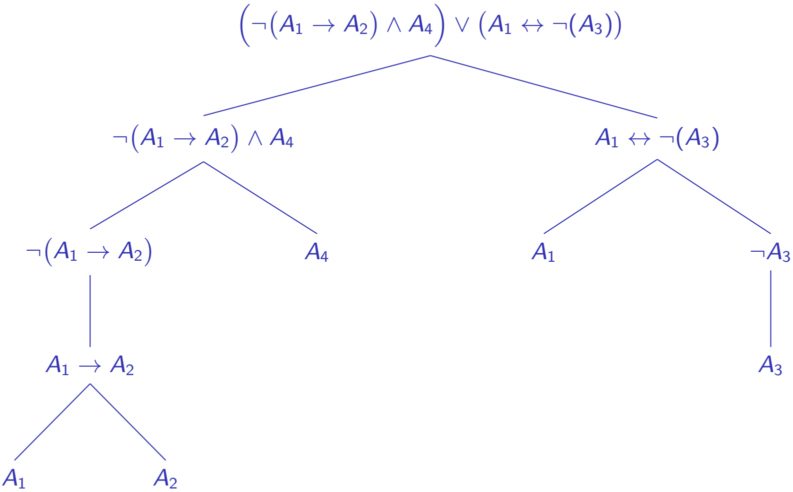
Is the following string a propositional sentence? Why?

$$\Big(\neg(A_1 \to A_2) \land A_4\Big) \lor (A_1 \leftrightarrow \neg(A_3))$$

How about

$$(\to A_1 A_2 \land (($$

Derivation Tree



Variables of a Propositional Sentence

Let ϕ be a propositional sentence. We define the variables operator recursively. $\text{Var}(\langle A_i \rangle) = \{A_i\}$ Now suppose $\text{Var}(\phi)$ is defined when the length (domain) of ϕ is less than m. Then we have one of the following cases:

- $\phi_i = \phi_j \wedge \phi_k, \quad \text{Var}(\phi_i) = \text{Var}(\phi_j) \cup \text{Var}(\phi_k)$
- $\phi_i = \phi_j \vee \phi_k, \quad \text{Var}(\phi_i) = \text{Var}(\phi_j) \cup \text{Var}(\phi_k)$
- $\phi_i = \phi_j \rightarrow \phi_k, \quad \text{Var}(\phi_i) = \text{Var}(\phi_j) \cup \text{Var}(\phi_k)$
- $\phi_i = \phi_j \leftrightarrow \phi_k, \quad \text{Var}(\phi_i) = \text{Var}(\phi_j) \cup \text{Var}(\phi_k)$
- $\phi_i = \neg\phi_j, \quad \text{Var}(\phi_i) = \text{Var}(\phi_j)$
- $\phi_i = (\phi_j), \quad \text{Var}(\phi_i) = \text{Var}(\phi_j)$

Truth Assignment for a finite set of Variables

In order to talk about the truth of certain sentences in propositional calculus, we define truth assignments for a given set of variables. A truth assignment is simply a binary function with its domain being a set of variables.

$\overline{x}$ is a truth assignment for $\{A_{i_1}, ..., A_{i_k}\} \iff \overline{x} : \{A_{i_1}, ..., A_{i_k}\} \to 2$

Example: if $\overline{x}(A_1) = 0, \overline{x}(A_2) = 1, \overline{x}(A_3) = 1, \overline{x}(A_6) = 0$, then $\overline{x}$ is a truth assignment on $\{A_1, A_2, A_3, A_6\}$. This could assign meaning to a propositional sentence like $((A_1 \wedge A_3) \to A_6) \vee A_2$

Basic Binary Operators

Now we define a few basic operators, these should agree with the common notions used in truth tables and basic logical constructions. All of these should be seen as functions from $\{0,1\}^2 \to \{0,1\}$.

$$f_\wedge(x,y) = xy, \quad f_\vee(x,y) = sgn(x+y), \quad f_\neg(x) = 1-x$$

Try to think of a function to express $f_\to(x,y)$!

Truth Tables

Mathematically, the truth table of a propositional sentence is a binary valued function on the set of truth assignments for the variables of that propositional sentence.

$$TT_\phi : 2^{Var(\phi)} \to 2$$

We define the truth table function recursively, since it is defined for propositional sentences which are themselves defined recursively.

- $TT_{\langle A_i \rangle}(\overline{x}) = \overline{x}(A_i)$
- If $\phi = \phi_1 @ \phi_2$, for @ a binary connective, then $TT_\phi(\overline{x}) = f_@\Big(TT_{\phi_1}(\overline{x}|_{Var(\phi_1)}), TT_{\phi_2}(\overline{x}|_{Var(\phi_2)})\Big)$
- If $\phi = \neg\phi_1$ then $TT_\phi(\overline{x}) = f_\neg\Big(TT_{\phi_1}(\overline{x}|_{Var(\phi_1)})\Big)$

Truth Table Example

Example: Calculate the Truth Table of $\phi = \neg((A_0 \wedge A_1) \rightarrow A_1)$

We begin by observing $\mathrm{Var}(\phi) = \{A_0, A_1\}$. This means there are $2^2 = 4$ truth assignments we must calculate to fully calculate the truth table of ϕ. We can write them out explicitly, the usual convention is to write the truth assignments out as in the table below.

A_0	A_1	$A_0 \wedge A_1$	$(A_0 \wedge A_1) \rightarrow A_1$	$\neg((A_0 \wedge A_1) \rightarrow A_1)$
1	1	1	1	0
1	0	0	1	0
0	1	0	1	0
0	0	0	1	0

Tautologies and Contradictions

When the truth table for a propositional sentence is the constant function 1, we call that propositional sentence a Tautology. Similarly when the truth table for a propositional sentence is the constant function 0, we call it a Contradiction.

For instance in the last slide we showed $\phi = \neg((A_0 \wedge A_1) \to A_1)$ is a contradiction by computation of its truth table.

The power of our binary functions and truth tables

The more mathematical scheme of definitions we have adopted allows us to prove many facts about general propositional sentences without computing individual truth tables as before. For example: If ϕ_1 and ϕ_2 are tautologies, prove $\phi_1 \wedge \phi_2$ is a tautology.

$$
\begin{aligned}
TT_{\phi_1 \wedge \phi_2}(\overline{x}) &= f_\wedge\Big(TT_{\phi_1}(\overline{x}|_{Var(x_1)}), TT_{\phi_2}(\overline{x}|_{Var(x_2)})\Big) \\
&= \Big(TT_{\phi_1}(\overline{x}|_{Var(x_1)}\Big)\Big(TT_{\phi_2}(\overline{x}|_{Var(x_2)}\Big) \\
&= (1)(1) = 1
\end{aligned}
$$

Repeated Conjunction and Disjunction

Let ϕ_i be a propositional sentence. We define recursively the following two operations.

$$\bigwedge_{i=0}^{0} \phi_i = \phi_0, \quad \bigwedge_{i=0}^{n+1} \phi_i = \left(\bigwedge_{i=0}^{n} \phi_i \right) \wedge \phi_{n+1}$$

$$\bigvee_{i=0}^{0} \phi_i = \phi_0, \quad \bigvee_{i=0}^{n+1} \phi_i = \left(\bigvee_{i=0}^{n} \phi_i \right) \vee \phi_{n+1}$$

Truth Tables of Repeated Conjunctions and Disjunctions

The decision to denote truth values by $\{0, 1\}$ rather than $\{T, F\}$ will now become very useful. We derive the truth table for the previously defined repeated conjunction and disjunction.

$$\begin{aligned}
TT_{\bigwedge_{i=0}^{n+1} \phi_i}(\overline{x}) &= TT_{(\bigwedge_{i=0}^{n} \phi_i) \wedge \phi_{n+1}}(\overline{x}) \\
&= f_\wedge\Big(TT_{(\bigwedge_{i=0}^{n} \phi_i)}(\overline{x}|_{Var(\bigwedge_{i=0}^{n} \phi_i)}), TT_{\phi_{n+1}}(\overline{x}|_{Var(\phi_{n+1})})\Big) \\
&= \Big(TT_{(\bigwedge_{i=0}^{n} \phi_i)}(\overline{x}|_{Var(\bigwedge_{i=0}^{n} \phi_i)})\Big) TT_{\phi_{n+1}}(\overline{x}|_{Var(\phi_{n+1})}) \\
&= \cdots = \prod_{i=0}^{n+1} TT_{\phi_i}(\overline{x}|_{Var(\phi_i)})
\end{aligned}$$

Continued

Similarly for repeated disjunctions, we apply the definition of repeated disjunction, the definition of $f_\vee$ and activate the inductive hypothesis.

$$\begin{aligned} TT_{\bigvee_{i=0}^{n+1} \phi_i}(\overline{x}) &= TT_{(\bigvee_{i=0}^{n} \phi_i) \vee \phi_{n+1}}(\overline{x}) \\ &= f_\vee\Big(TT_{(\bigvee_{i=0}^{n} \phi_i)}(\overline{x}|_{Var(\bigvee_{i=0}^{n} \phi_i)}), TT_{\phi_{n+1}}(\overline{x}|_{Var(\phi_{n+1})})\Big) \\ &= sgn\Big(TT_{(\bigvee_{i=0}^{n} \phi_i)}(\overline{x}|_{Var(\bigvee_{i=0}^{n} \phi_i)}) + TT_{\phi_{n+1}}(\overline{x}|_{Var(\phi_{n+1})})\Big) \\ &= \cdots = sgn\Big(\sum_{i=0}^{n+1} TT_{\phi_i}(\overline{x}|_{Var(\phi_i)})\Big) \end{aligned}$$

Directed Reading Text

A Mathematical Introduction to Logic by Herbert Enderton