A Conceptual Tour Through Frederic Schuller's *Lectures on the Geometrical Anatomy of Theoretical Physics*

Siddhartha Bhattacharjee
Website: Tempus Spatium
Email: sidb@tempus-spatium.8shield.net

Student, B.Sc. (Hons) Mathematical Physics, University of Waterloo

March 31, 2023

Abstract

In these notes, I venture through Dr. Frederic P. Schuller's *Lectures on the Geometrical Anatomy of Theoretical Physics*, found on YouTube. This venturing entails interpreting his construction of ideas and discussing them both formally and informally. Inspiration for the structure of the notes is drawn from Simon Rhea's notes on the lectures.

The aim of this project is to capture and add to some of the intuition in the lectures, in the form of proofs, remarks and other ideas. This will be done in a manner which fits pedagogically and chronologically into both Dr. Frederic Schuller's lectures and Simon Rhea's notes.

The manner of writing these notes is fairly personal. That said, its construction will hopefully allow it to be, for some, a useful resource that can be used alongside the original lectures and notes based on them.

Contents

1 Logic of propositions and predicates

1.1 Propositional logic

Definition 1.1. *A **proposition** p is a formal expression which can take the forms* true *(T) and* false *(F).*

Informally[1], propositional logic is the algebra of propositions. By 'algebra', we mean the *compositionality* of objects (here propositions) via operators (here, logical connectives).

Definition 1.2. *A **tautology** is a proposition that is always T. A **contradiction** is a proposition that is always F.*

It is tempting to think of logical operators/connectives as maps from one or more propositions, to a new proposition. However, as we are yet to construct set theory, such a treatment would be logically circular[2]. Therefore, the best we can do to define a logical operator is to lay out what it does to every combination of its input propositions. Think of this as 'fleshing out' the rules characterizing the said logical operator. Typically, this 'fleshing out' is done using truth tables.

If a logical operator $\mathcal{L}$ acts on n propositions, there are $2^{(2^n)}$ ways of defining it. This corresponds to 2^n entries in the truth table for $\mathcal{L}$, each of which corresponds to some unique combination of entries whose output is fixed to have some truth value from the two choices, T and F.

Following are truth tables and discussion of some common logical operators.

[1] As we have not yet formally constructed notions such as that of an algebra, any allusion to such structures is only for informal and intuitive purposes.

[2] A possible way to get around this is to construct categorical logic. However, category theory may be based on some parts of set theory, depending on whom one asks. In particular, category theory uses the notion of classes, which are set-like objects although they may not obey set-theoretic axiomatic systems such as the ZFC axioms.

1.1.1 Unary operators

Unary operators are logical connectives which act on one proposition and produce another. Thus, there are $2^{\left(2^1\right)} = 4$ possible unary operators, listed below:

p	id_p	$\neg p$	$\bot p$	$\top p$
F	F	T	F	T
T	T	F	F	T

where id is the *identity* operator, $\neg$ is the *negation* operator, $\bot$ is the *contradiction* operator and $\top$ is the *tautology* operator.

1.1.2 Binary operators

There are $2^{\left(2^2\right)} = 16$ binary logical connectives. Some particularly important ones are:

p	q	$p \wedge q$	$p \vee q$	$p \veebar q$	$p \implies q$	$p \iff q$
F	F	F	F	F	T	T
F	T	F	T	T	T	F
T	F	F	T	T	F	F
T	T	T	T	F	T	T

The above binary operators are respectively called the *and*; *or*; *exclusive or*; *implication*; and *equivalence* operators.

Remark 1.1 (Binding strength)**.** *In a given formal expression, the binary logical operator with the highest* binding strength *separates the expression into two smaller expressions (typically on either side of the said binary operator). The expression can thus be converted to an equivalent expression by making the binary operator act on the mentioned expressions. This process can then be repeated for the smaller expressions, provided the binary operators appearing in them, if any, have a lower binding strength than the previous one. This forms a recursive process which is an example of* evaluating *the original formal expression.*

Remark 1.2 (Principle of explosion)**.** *The idea that $F \implies T$ is called the* principle of explosion *or* ex falso quodlibet*, i.e. from a false assumption, anything true can follow. The intuition for this principle is that once a contradiction has been established, any proposition, including negations, can be gleaned from the contradiction.*

This results in the following important theorem.

Theorem 1.1 (Contraposition). *If p and q are propisitions, then,*

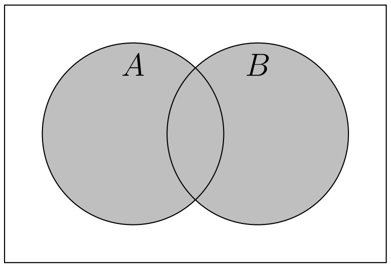
$$(p \implies q) \iff (\neg q \implies \neg p)$$

Proof. We observe the following truth table, which by the definition of the equivalence operator, helps establish the above result:

p	q	$p \implies q$	$\neg q \implies \neg p$	$(p \implies q) \iff (\neg q \implies \neg p)$
F	F	T	T	T
F	T	T	T	T
T	F	F	F	T
T	T	T	T	T

□

A clue into the truth of the above theorem can be seen in the following Venn diagrams[3].



Theorem 1.2 (De Morgan's Laws). *De Morgan's laws or theorem is a set of the following two rules,*

$$\neg (p \wedge q) = \neg p \vee \neg q$$

$$\neg (p \vee q) = \neg p \wedge \neg q$$

The above statements can be exhaustively shown to be true using truth tables.

[3]At this stage, Venn diagrams are only intuitive tools but yet to be constructed rigorously as we have not yet defined sets, on which these diagrams are based.

Definition 1.3. *The **nand** operator $\uparrow$ is the binary operator defined by:*

p	q	$p \uparrow q$
F	F	T
F	T	T
T	F	T
T	T	F

We have that $p \uparrow q = \neg (p \wedge q)$. This shows how the *nand* operator internally contains the *not* and *and* operators.

Lemma 1.1 (Disjunctive normal form)**.**

Proof. □

Theorem 1.3 (Functional completeness)**.** *Any N-ary logical operator* $\heartsuit (p_1, \ldots, p_N)$ *can be constructed using the* nand *($\uparrow$) binary operator.*

Proof. □

Example 1.1.