1- Truth Table

As an example of using the truth table for ∧, suppose you know that both ***p*** and ***q*** are ***T***. Look in the truth table for ∧ to find the row where both ***p*** and ***q*** have the value ***T***. Then, look across that row to find the truth value of $p \wedge q$. In this case, $p \wedge q$ has the value ***T***. Now, suppose in another instance you know that ***p*** is ***T*** and ***q*** is ***F***. The second row of the table for ∧ has the value ***T*** for ***p*** and ***F*** for ***q***. In that row, the truth value given for $p \wedge q$ is ***F***.

It is helpful to consider how the truth table for → relates to common usage of "if... then." A simple requirement of a notion of "if ... then" is that "if ... then" statements should be usable in arguments. If it is true that "The carriage had mud on its tires" and is also true that "If the carriage had mud on its tires, then it is raining outside," then one can correctly infer that "It is raining outside." The truth table definition of → is that $p \rightarrow q$ is ***F*** just in case it would lead from a true hypothesis to a false conclusion.

2- Formulas

More complicated propositional expressions, called **formulas** or**well-formed formulas** (**wffs**), can be built from the proposition letters using the propositional connectives and parentheses. When we say $\emptyset = (p \wedge q) \rightarrow r$, we mean that $\emptyset$ is the string of symbols $\emptyset = (p \wedge q) \rightarrow r$. For the following formulas, we would like to know when the conclusion is necessarily true:

a. $\emptyset = (p \wedge q) \rightarrow r$, which can be paraphrased as "If ***p*** and ***q*** are both true, then ***r*** is also true."
b. $\emptyset 1 = (p \vee q) \rightarrow r$, which can be paraphrased as "If ***p*** or ***q*** (or both) is true, then ***r*** is also true."
c. $\emptyset 2 = (p \rightarrow r) \rightarrow ((p \wedge q) \rightarrow r)$, which can be paraphrased as "Suppose that if ***p*** is **T**, then ***r*** is **T** Then, if ***p*** and ***q*** are both **T**, then ***r*** is **T**."

In the last formula, we translated two of the → 's as if... then and one as suppose ... then. We did that to make the reading easier. One advantage of formal notation is that it lets us express concepts that cannot be expressed easily and unambiguously in everyday language.

Example 3. Translate the following sentences into a formula in propositional logic: "If Mr. Holmes told the truth and Mr. Watson did not hear anything, then it cannot be both that the butler did it and that the butler returned to his hotel room that night."

Solution.

Actually, there are many translations, depending on which parts of the sentence are chosen to be represented by proposition letters and on which proposition letters are chosen to represent them. Let $\boldsymbol{p}$ denote "Mr. Holmes told the truth, "$\boldsymbol{q}$ denote "Mr. Watson did not hear anything," $\boldsymbol{r}$ denote "the butler did it," and $\boldsymbol{s}$ denote "the butler returned to his room that night." The sentence can now be translated into propositional logic as

$$\emptyset = (\boldsymbol{p} \wedge \boldsymbol{q}) \rightarrow (\neg(\boldsymbol{r} \wedge \boldsymbol{s}))$$

3- Expression Trees for Formulas

An expression tree is simply a visual representation for the way that a formula is built from propositions and logical operators. A proposition is represented by a single node, simply a filled-in circle, as shown in **Figure 1.1**.

Figure 1.1. Representation for $\boldsymbol{p}$.

For an expression involving two propositions and a logical operator, the propositions are represented by nodes at the same level, and then at a higher level, a node represents the result of applying the operator to the two propositions. The nodes representing the propositions and the node representing the result of the operation are joined by lines. For example, the final picture for p v q is shown in **Figure 1.2**.

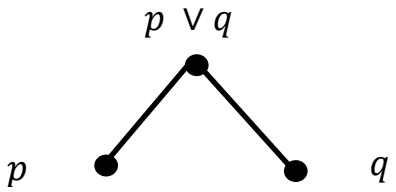


Figure 1.2. Representation for p v q.

To introduce the representation structure for a more general formula, we will describe how you build an expression tree from the top down. To build an expression tree from an expression, first place the final expression at the top of the representation, and then put the expressions that are operated on to form the final expression underneath. Join by lines the nodes representing the expressions operated on and the node representing the result of the operation. The process can continue until the lowest level contains only propositions. The resulting picture or representation of an expression is an expression tree.

The expression tree structure gives exactly the same information as the parentheses in the formula about the order of execution, but the expression tree sometimes gives a better picture. Because this representation is so useful in evaluating an expression, we will give several more examples and then a formal description of how you can build an expression tree from the bottom up. The expression tree of ((p ∧ q) ∧ r) is shown in **Figure 1.3**.

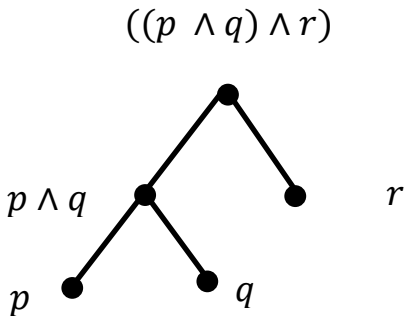


Figure 1.3. Representation for $((p \wedge q) \wedge r)$

The expression tree of tree of $((\neg p) \vee q) \rightarrow (r \rightarrow p)$ is shown in **Figure 1.4**.

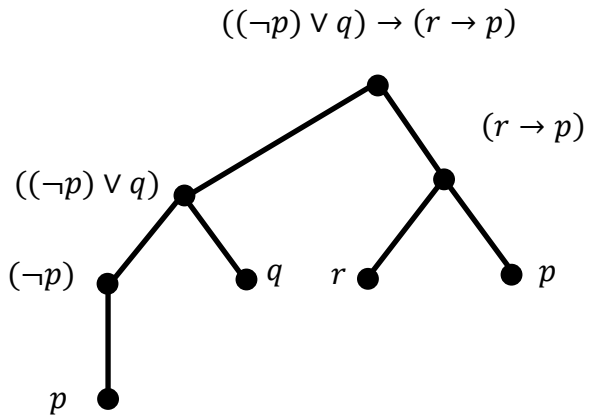
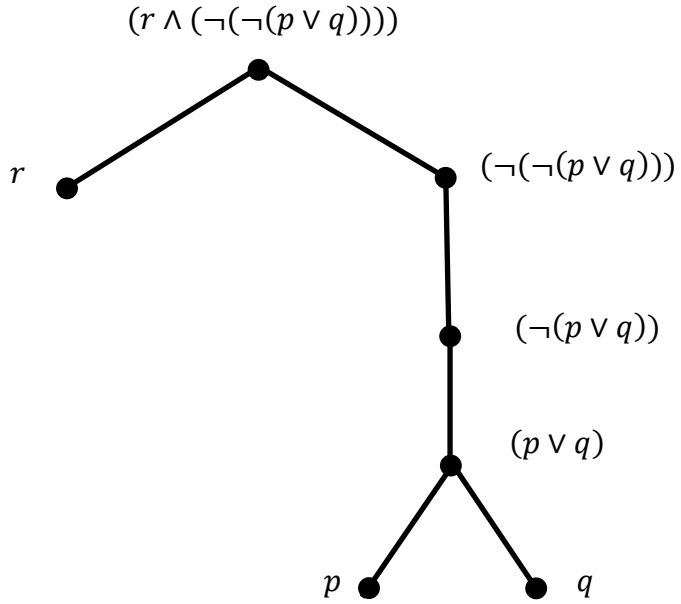


Figure 1.4. Expression tree of $((\neg p) \vee q) \rightarrow (r \rightarrow p)$

Example 4. For the expression tree T, determine the sub-formulas defined by ***p*** and $(\neg(p \vee q)$.



Solution. The subtrees T_p and $T_{(\neg(p \vee q))}$ are as shown: