Handout 4

PL: Truth Trees

Truth tables provide a mechanical method for determining whether a proposition, set of propositions, or argument has a particular logical property. For example, we can show that an argument is deductively valid (or invalid) using the truth-table method. The problem with truth tables is that they are cumbersome when dealing with formulas that have more than two propositional letters. While a truth table for a proposition with two propositional letters only has four rows, a truth table for a proposition with three propositional letters has eight, a proposition with four letters has sixteen, and so on.

In this lesson, an alternative decision procedure is articulated. This is the truth-tree method. The advantage of truth trees is that it is a decision procedure whose complexity is not a function of the number of propositional letters in the formula being analyzed. And, as we will see later, the method is capable of being adapted to a more expressive logical language.

4.1 Truth Trees: The Setup

Truth trees first begin with an initial setup involving three columns:

1. for numbering the propositions,
2. writing (stacking) the propositions,
3. justification of propositions.

To illustrate, consider the following two propositions: $P \wedge R, M \wedge \neg P$

column 1	column 2	column 3
1	$P \wedge R$	P
2	$M \wedge \neg P$	P

Table 4.1 – Setting up a truth tree: putting the propositions in a stack

In Ch.4 of Agler's *Symbolic Logic*, there is special attention paid to making sure that all trees cite the justification of each of the rows of the tree. In this handout, much of the justification for trees is not provided.

4.2 Truth Trees: Some Basics in Decomposition (introducing $\wedge D$)

Next, propositions are decomposed (broken apart) according to the conditions under which they are true. Decomposition occurs by applying specific decomposition rules. There are 9 decomposition rules, each applying to a specific type of decomposable proposition (see Agler *Symbolic Logic*, p.86). For example, conjunctions can only be decomposed using conjunction decomposition ($\wedge D$) and not any other decomposition rule.

When you decompose a proposition, you write the decomposed propositions under the stack, you number these propositions, and you cite the decomposition rule you used (along with the line number of the proposition you are decomposing). As a helpful convention, you may want to place a check mark (✓) by the proposition you decomposed to indicate it has been decomposed. Before continuing any further, it will be helpful to look at Agler, *Symbolic Logic*, pp.105-107. There there is the formulation of the $\wedge D$ rule and some very simple examples.

Here, let's turn a slightly more complicated tree, one that involves multiple uses of $\wedge D$. Consider the following set of propositions: $P \wedge R, M \wedge \neg P$. First, we set up the tree (see Figure 4.1).

1	$P \wedge R$
2	$M \wedge \neg P$

Figure 4.1 – Truth tree setup.

Next, we begin the process of decomposing the tree. Let's start by decomposing $P \wedge R$ (see Figure 4.2).

1	$P \wedge R$✓
2	$M \wedge \neg P$
3	P
4	R

Figure 4.2 – Truth tree where $P \wedge R$ has been decomposed using $\wedge D$

Notice, however, that $M \wedge \neg P$ is a conjunction and so it too can be decomposed using $\wedge D$ (see Figure 4.3).

The tree is now fully decomposed.

1	$P \wedge R\checkmark$
2	$M \wedge \neg P\checkmark$
3	P
4	R
5	M
6	$\neg P$

Figure 4.3 – Truth tree where $M \wedge \neg P$ has been decomposed using $\wedge D$

.

4.3 Truth Trees: Stacking vs. Branching (introducing $\vee D$)

There are three major kinds of decomposition rules: stacking, branching, and branching and stacking. When decomposing a proposition P, a stacking rule represents that there is one condition in which P is true, a branching rule represents that there are three conditions in which the P is true, and a branching and stacking rule represents that there are two conditions in which P is true. All of these rules look different. Consider the following propositions: $P \wedge R, M \vee \neg P$

1	$P \wedge R\checkmark$	
2	$M \vee \neg P$	
3	P	$1 \wedge D$
4	R	$1 \wedge D$

Since $P \wedge R$ (at line 1) is a conjunction, and $P \wedge R$' is true if P is true and R is true, we can represent this by stacking P and R under the existing set of propositions. In contrast, $M \vee \neg P$ is a disjunction and is true if either M is true or $\neg P$ is true. This is represented by creating two branches (or paths) that break from the stack of propositions.

1	$P \wedge R\checkmark$		
2	$M \vee \neg P\checkmark$		
3	P		
4	R		
5	M ↙	↘ $\neg P$	$2 \vee D$

4.4 Truth Trees: Some Terminology

In this section, some terminology is developed for talking about truth trees.

Definition – Branch

All the propositions obtained by starting from the bottom of the tree and reading upward through the tree.

Definition – Fully decomposed branch

A branch is fully decomposed when all propositions in the branch that can be decomposed have been decomposed.

Definition – Partially decomposed branch

A branch is partially decomposed when there is at least one proposition in the branch that has not been decomposed.

Definition – Closed Branch

A branch is closed when it contains a proposition P and its literal negation $\neg(P)$. A closed branch is represented by an ✗.

Definition – Open Branch

An open branch is a branch that is not closed. That is, a branch that does not contain a proposition P and its literal negation $\neg(P)$.

Definition – Completed Open Branch

A completed open branch is a fully decomposed branch that is not closed. That is, it is a fully-decomposed branch that does not contain a proposition P and its literal negation $\neg(P)$. An open branch is denoted by writing an **O** under the last proposition in the open branch.

Definition – Completed Open Tree

A tree is a completed open tree if and only if it has at least one completed open branch. That is, a tree is a completed open tree if and only if it contains at least one fully decomposed branch that is not closed. A completed open tree is a tree where there is at least one branch that has an **O** under it.

Definition – Closed Tree

A tree is closed when all of the tree's branches are closed. A closed tree will have an ✗ under every branch.

Definition – Descending Decomposition Rule

When decomposing a proposition P, decompose P under every open branch that descends from P.

Let's consider a few examples. First, consider the tree in Figure 4.4

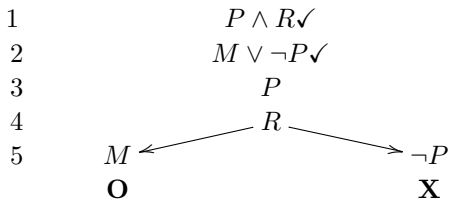

Figure 4.4 – Example #1: Basic Terminology

Lines 1 and 2 form the tree's **stack** or the propositions composing the initial setup of the tree. The tree has two branches. Starting from the left-hand side with M at line 5, we read upward and identify $M, R, P, M \vee \neg P, P \wedge R$ as belonging to one branch. See Figure 4.5. Second, beginning with the $\neg P$ at the right-hand side at the bottom, we read upward and identify the formulas in the second branch. See Figure 4.6

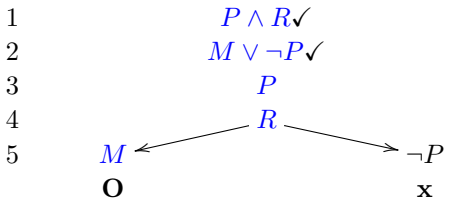

Figure 4.5 – Branch 1: $M, R, P, M \vee \neg P, P \wedge R$

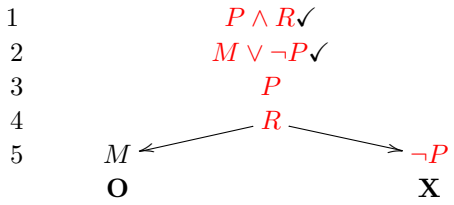

Figure 4.6 – Branch 2: $\neg P, R, P, M \vee \neg P, P \wedge R$.

Branch #1 is a **completed branch** as all of the propositions that can be decomposed have been decomposed (more on this in the next section). In addition,

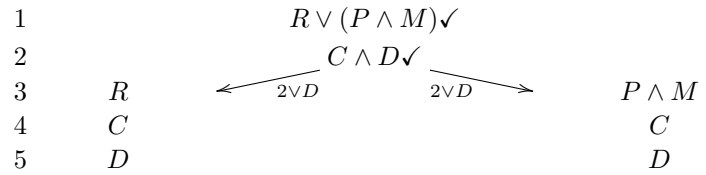

Figure 4.7 – Initial Tree

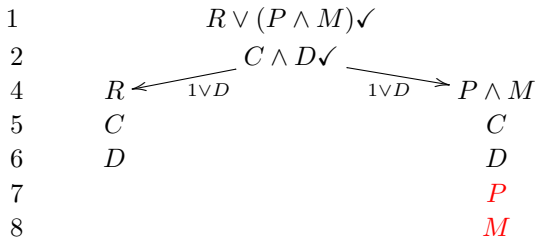

Figure 4.8 – Correct Use of Descending Decomposition Rule

it is a **completed open branch** as there is no proposition **P** and its literal negation $\neg$(**P**) in $M, R, P, (M \vee \neg P), (P \wedge R)$.

Branch #2 is also a **completed branch**, but it is a **closed branch** as there is an instance of the proposition P and $\neg P$ in the branch $\neg P, R, P, M \vee \neg P, P \wedge R$.

As the tree has one completed open branch, the tree is classified as an **open tree**. If all of the branches were closed, the tree would be **closed**.

The **descending decomposition rule** states that when decomposing a proposition P, decompose P under every open branch that descends from P (see Agler, *Symbolic Logic* (1st ed.) pp.100–103). This means that when you decompose P you decompose it under every open branch that descends from P. You are not required to decompose P under closed branches and should not decompose it under branches that do not descend from P.

When considering the tree above, there is a question of whether $P \wedge M$ should be decomposed under **both** branches or **only under the right branch**. If we follow the descending decomposition rule, we should only decompose $P \wedge M$ under the **right branch** since the left branch is not a branch that descends from $P \wedge M$.

4.5 Truth Trees: Remaining Decomposition Rules

A fully decomposed branch was defined as a branch where all propositions in the branch that can be decomposed have been decomposed. However, in putting

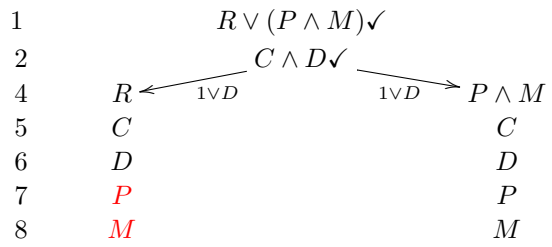

Figure 4.9 – Incorrect Use of Descending Decomposition Rule

forward this definition, no mention was made of which propositions can be decomposed. Any wff that is neither a propositional letter P or its literal negation $\neg(P)$ is decomposable. For our purposes, however, wffs are grouped into *nine* different proposition types and there is a decomposition rule for each one of these types.

Proposition Types	Wff	D-Rule
conjunction	$(P \wedge Q)$	$\wedge D$
disjunction	$(P \vee Q)$	$\vee D$
conditional	$(P \rightarrow Q)$	$\rightarrow D$
biconditional	$(P \leftrightarrow Q)$	$\leftrightarrow D$
negated conjunction	$\neg(P \wedge Q)$	$\neg \wedge D$
negated disjunction	$\neg(P \vee Q)$	$\neg \vee D$
negated conditional	$\neg(P \rightarrow Q)$	$\neg \rightarrow D$
negated biconditional	$\neg(P \leftrightarrow Q)$	$\neg \leftrightarrow D$
double-negation	$\neg(\neg P)$	$\neg\neg D$

Table 4.2 – Decomposable Proposition Types and Decomposition Rules

Choosing which decomposition rule to apply to a proposition is entirely determined by the type of proposition it is. If **P** is a conjunction, then you apply the corresponding decomposition rule for conjunctions: $\wedge D$. If **P** is a disjunction, you apply $\vee D$ as this is the decomposition rule for disjunctions. And so on.

Exercise 1: *Create a truth tree for the following sets of wffs, then determine if the tree is open or closed:*

1. $P \wedge \neg Q$
2. $P \wedge Q, P \rightarrow Q, P \vee Q$
3. $\neg P \vee \neg Q, \neg(P \wedge Q), P \leftrightarrow Q$

4.6 Truth Trees: Decomposition Strategies

Using the decomposition rules blindly ultimately lead to a completed open or a closed tree. For example, for a tree involving $P \wedge Q, P \rightarrow R$, it is equally

$P \wedge Q$ P Q	 $\wedge D$ $\wedge D$	$P \vee Q$ P $\quad$ Q $\quad$ $\vee D$
$\neg(P \vee Q)$ $\neg(P)$ $\neg(Q)$	 $\neg \vee D$ $\neg \vee D$	$\neg(P \wedge Q)$ $\neg(P)$ $\quad$ $\neg(Q)$ $\quad$ $\neg \wedge D$
$\neg(P \rightarrow Q)$ P $\neg(Q)$	 $\neg \rightarrow D$ $\neg \rightarrow D$	$P \leftrightarrow Q$ P $\quad$ $\neg(P)$ $\quad$ $\leftrightarrow D$ Q $\quad$ $\neg(Q)$ $\quad$ $\leftrightarrow D$
$\neg\neg(P)$ P	 $\neg\neg D$	$\neg(P \leftrightarrow Q)$ P $\quad$ $\neg(P)$ $\quad$ $\neg \leftrightarrow D$ $\neg(Q)$ $\quad$ Q $\quad$ $\neg \leftrightarrow D$

Table 4.3 – Truth tree decomposition rules for **PL**

acceptable to begin by decomposing either of the two propositions. However, a **strategic use** of these rules will lead to the same result in a timelier manner. There are four such strategic rules:

Strategic Rule 1: Use no more rules than needed.
Strategic Rule 2: Use rules that close branches.
Strategic Rule 3: Use stacking rules before branching rules.
Strategic Rule 4: Decompose more complex propositions before simpler propositions.

Table 4.4 – Decomposition Strategies for **PL** Truth Trees

Keep in mind that these rules are "rules of thumb" and so the successful decomposition of a tree does not require you to follow them.

Exercise 2: *Use a truth tree to determine if the tree is open or closed. To simplify your work, use the strategic rules (Note: you may not need to decompose every wff to determine if the tree is open or closed).*

1. $P \vee (Q \vee R), P \wedge \neg P$
2. $(P \vee P) \wedge (R \wedge \neg R), S \leftrightarrow T, P \vee M$
3. $(P \wedge Q) \wedge R, M \vee (Q \vee \neg R)$

4.7 Truth Trees: Analysis

Truth trees can be used to determine various semantic properties about propositions, sets of propositions, and arguments. Using truth trees to do this requires that you (i) set up the tree in a specific way to test for a specific property (you can't just stack the propositions in every instance), (ii) know whether a closed

(or completed open) tree indicates a specific semantic property.

4.7.1 Recovering an Interpretation

From a completed open branch in a truth tree, we can construct an interpretation (an assignment of T or F to propositional letters) that would make all of the wffs in that branch true. For example, consider the tree in Figure 4.10.

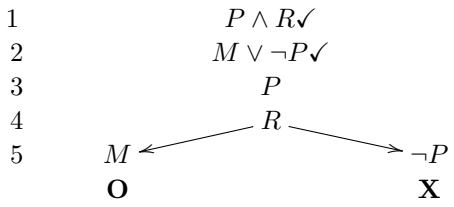

Figure 4.10 – Recovering an interpretation from a tree for $P \wedge R, M \vee \neg P$

In order to recover an interpretation from an open branch: for every atomic wff in the branch, assign a T to the corresponding propositional letter, and for every negated atomic wff, assign F to the propositional letter contained in that formula. Alternatively, you can sign T to the negated wff. So, since M, R, P are atomic wffs in the open branch of Figure 4.10, an interpretation is recovered from Figure 4.10 by assigning $\mathscr{I}(M) = T, \mathscr{I}(R) = T, \mathscr{I}(P) = T$.

4.7.2 Sets of Propositions: Consistency

A completed open branch tells us that there is a way to assign truth values (an interpretation) that would make all of the wffs in the branch true. Given this is the case, a completed open branch tells us that the stack of wffs (and all of the wffs in the branch for that matter) are consistent.

> Definition – Consistent
>
> A set of propositions $\{P, Q, R, \ldots Z\}$ is consistent if and only if there is at least one interpretation where $P, Q, R, \ldots Z$ are true. A truth tree shows that $\{P, Q, R, \ldots, Z\}$ is consistent if and only if a complete tree of the stack of $P, Q, R, \ldots, Z$ determines a completed open tree. That is, if there is at least one completed open branch.

In contrast, if a truth tree for a set of wffs yields a closed tree. That is, a truth tree where there is not a completed open branch, then there is no way to assign truth values to the propositional letters (an interpretation) that would make the propositions in the stack come out true. In other words, a closed tree tells us that the stack of wffs are inconsisent.

DEFINITION – INCONSISTENT

A set of propositions $\{P, Q, R, \ldots Z\}$ is inconsistent if and only if there is no interpretation where $P, Q, R, \ldots Z$ are true. A truth tree shows that $\{P, Q, R, \ldots, Z\}$ is inconsistent if and only if a tree of the stack of $P, Q, R, \ldots, Z$ is a closed tree. That is, if all branches close.

Suppose you are testing whether the set $\{P \vee Q, Z \vee R\}$ is consistent / inconsistent. Begin by writing each proposition on a separate line in the tree.

1	$P \vee Q$
2	$Z \vee R$

Figure 4.11 – Setup to test whether $P \vee Q, Z \vee R$ are consistent.

Next, we decompose the tree.

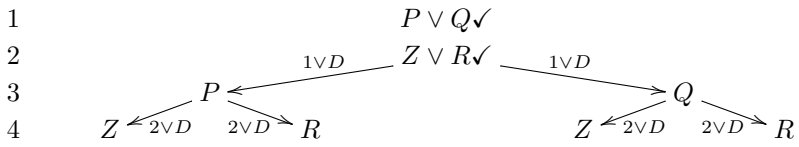

Figure 4.12 – A tree with at least one completed open branch showing that $P \vee Q$ and $Z \vee R$ are consistent.

Finally, we analyze or "read" the tree by looking to see whether the tree has a completed open branch. Since it does, the truth-tree test reveals that $\{P \vee Q, Z \vee R\}$ is **consistent**, viz., we can assign truth values to P, Q, and Z such that $P \vee Q$ and $Z \vee R$ both come out as true.

In contrast, the truth tree for $\{\neg(S \to T), \neg\neg(\neg S \vee T)\}$ shows that that set is inconsistent.

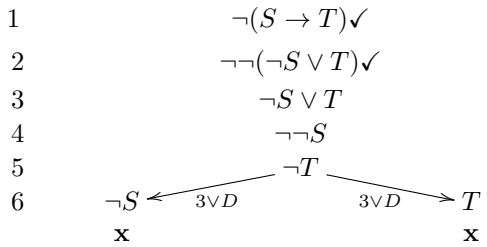

Figure 4.13 – A closed tree that shows $\neg(S \to T)$ and $\neg\neg(\neg S \vee T)$ are inconsistent.

Starting from the base of the tree, notice that there is a proposition **P** and its literal negation $\neg(\mathbf{P})$ in both branches. In the left branch, there is $\neg S$ and S while in the right branch there is T and $\neg T$. As such, there is no way we can give a uniform assignment of truth values to the propositional letters that compose the stack that would make $\{\neg(S \to T), \neg\neg(\neg S \vee T)\}$ true.

4.7.3 Propositions: Tautology, Contradiction, Contingency

Definition – Tautology

A proposition P is a tautology if and only if P is true under every valuation. A truth tree shows that P is a tautology if and only if a tree of the stack of $\neg(P)$ determines a closed tree.

Definition – Contradiction

A proposition P is a contradiction if and only if P is false under every valuation. A truth tree shows that P is a contradiction if and only if a tree of the stack of P determines a closed tree.

Definition – Contingency

A proposition P is a contingency if and only if P is neither always false under every valuation nor always true under every valuation. A truth tree shows that P is a contingency if and only if a tree of P does not determine a closed tree and a tree of $\neg(P)$ does not determine a closed tree.

4.7.4 Pairs of Propositions – Equivalence

Definition – Equivalence

A pair of propositions P, Q is equivalent if and only if P and Q have identical truth values under every valuation. A truth tree shows that P and Q are equivalent if and only if a tree of $\neg(P \leftrightarrow Q)$ determines a closed tree.

1 $\neg((P \to Q) \leftrightarrow (\neg P \vee Q))$

Figure 4.14 – Correct setup to test whether $P \to Q$ and $\neg P \vee Q$ are equivalent.

4.7.5 Arguments: Validity

Definition – Validity

An argument $P, Q, \ldots, Y$ therefore Z is valid in **PL** if and only if $P, Q, \ldots, Y \models Z$. A truth tree shows that an argument $P, Q, \ldots, Y \models Z$ if and only if $P, Q, R, \ldots, Y, \neg(Z)$ determines a closed tree.

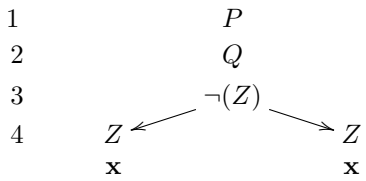

Figure 4.15 – Example of a tree that shows **P, Q** therefore **Z** is valid.

4.8 Truth Trees: What's the Point?

In lessons 1 and 3, we formulated two different methods for determining whether an argument was valid or invalid. In lesson 1, we used what might be called the "imagination test", a test that requires us to test the validity of the argument by trying to imagine a scenario where the premises are true and the conclusion is false. If we can imagine such a scenario, then the argument is **invalid**. If we can't, then the argument is said to be **valid**. In lesson 3, we pointed out two problems with this test. First, the test breaks down when dealing with large arguments as all of the states of affairs represented by the many English sentences are hard to keep in one's mind. Second, the test is subject to human bias.

After we developed the logical language PL in lesson 3, we attempted to replace the "imagination test" with the "truth-table test." The primary merit of truth tables were that they could be used to determine whether certain arguments were valid or invalid in a *completely mechanical way*. Truth tables thus introduce an automated method of checking arguments, one that could avoid problems of human bias. However, we also noted that the truth-table method had its own problems. First, it could not easily handle arguments that involved many propositional letters. Second, we cannot represent **every** valid English argument in PL.

In this lesson, we formulated the truth-tree test. The primary merit of this method is that the complexity of truth trees is not a function of the number of propositional letters in the argument. That is, where the argument $\neg(M \vee S), \neg((T \vee R) \vee Q), \neg(\neg(S \vee R) \vee Q) \models \neg A$ would require a whopping **64 rows** ($2^6 = 64$), a truth tree quickly reveals the above argument is valid:

However, truth trees suffer from the same problem as truth tables. Since not every valid argument in English can be represented in PL, the truth-tree method can only tell us whether arguments that can be represented in PL are valid

1	$\neg(M \vee S)\checkmark$	P
2	$\neg((T \vee R) \vee Q)\checkmark$	P
3	$\neg(\neg(S \vee R) \vee Q)\checkmark$	P
4	$\neg\neg A\checkmark$	P
5	A	$4DN$
6	$\neg M$	$1\neg \vee D$
7	$\neg S$	$1\neg \vee D$
8	$\neg(T \vee R)$	$2\neg \vee D$
9	$\neg Q$	$2\neg \vee D$
10	$\neg\neg(S \vee R)\checkmark$	$3\neg \vee D$
11	$\neg Q$	$3\neg \vee D$
12	$S \vee T\checkmark$	$10DN$
13	$\neg T$	$8\neg \vee D$
14	$\neg R$	$8\neg \vee D$
15	S $\quad$ R	$12 \vee D$

Figure 4.16 – Truth tree for $\neg(M \vee S), \neg((T \vee R) \vee Q), \neg(\neg(S \vee R) \vee Q) \models \neg A$

or invalid. It does not tell us whether arguments that cannot be accurately represented in PL are valid or invalid. In lesson 6, we formulate a new, more expressive logical language that aims to incorporate more valid arguments in English. But, for now, in lesson 5, we turn to a discussion of how to formulate a set of inference or derivation rules that will allow us to show that a conclusion follows from a set of premises.

Questions

1. What is a branch in a truth tree?
2. What is a fully decomposed branch in a truth tree?
3. What is the difference between a closed branch and a completed open branch?
4. Can a branch be a completed open branch and a closed branch?
5. If a branch has a wff P and that wff's literal negation $\neg(P)$, is that branch closed or open?
6. What is a completed open tree?
7. Can a completed open tree have a closed branch?
8. What is a closed tree?
9. Can a closed tree have an open branch?
10. What is the descending decomposition rule?
11. Decompose a tree for $P \rightarrow \neg Q$. What does that completely decomposed tree tell us about the conditions under which $P \rightarrow \neg Q$?
12. Can $\wedge D$ be used to decompose conjunctions?
13. What type of wff does $neg \leftrightarrow D$ decompose?

14. Recover an interpretation from a tree whose stack consists of $\neg P \rightarrow Q, W \wedge S$.
15. Can an interpretation be recovered from a closed branch?
16. Determine whether $P \wedge \neg Q, P \vee \neg Q$ is consistent.
17. Determine whether $P \wedge \neg Q, S \vee \neg(M \vee R)$ is consistent.
18. Determine whether $P \rightarrow Q, P \vee Q$ is equivalent.
19. Determine whether $P \wedge P \vee P$ is a contingency, tautology, or contradiction.
20. Determine whether $P \rightarrow (Q \rightarrow P)$ is a contingency, tautology, or contradiction.
21. Determine whether $P \vee Q, P \rightarrow R, Q \rightarrow R$ therefore R is a valid argument.
22. In using a truth-tree test to determine whether $\mathbf{P}, \mathbf{Q}, \mathbf{R} \models \mathbf{Z}$, why is it the case that a closed tree tells us that $\mathbf{P}, \mathbf{Q}, \mathbf{R} \models \mathbf{Z}$ rather than $\mathbf{P}, \mathbf{Q}, \mathbf{R} \not\models \mathbf{Z}$
23. If $\mathbf{P} \rightarrow \mathbf{Q}$ is a tautology, is it the case that $\mathbf{P} \models \mathbf{Q}$? That is, if a conditional $\mathbf{P} \rightarrow \mathbf{Q}$ is a tautology, does the antecedent of that conditional $\mathbf{P}$ entail the consequent $\mathbf{Q}$?
24. If $\mathbf{P} \models \mathbf{Q}$, then is $\mathbf{P} \rightarrow \mathbf{Q}$ a tautology? That is, if $\mathbf{P}$ entails $\mathbf{Q}$, then is the conditional $\mathbf{P} \rightarrow \mathbf{Q}$ a tautology?
25. If the set of wffs $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ is inconsistent, then does $\mathbf{P}, \mathbf{Q}, \mathbf{R} \models \mathbf{Z}$ and $\mathbf{P}, \mathbf{Q}, \mathbf{R} \models \neg(\mathbf{Z})$?