Chapter 3

Truth Functional Logic: Determining Validity and Satisfiability by Calculation

3.1 Boolean Semantics

We define the notion of truth functional validity and truth functional entailment in 3.1.1, and in the next section, we redefine these concepts in terms of truth functions calculated by means of truth tables.

We introduce truth tables and Boolean vectors in 3.1.2. Truth tables give the output of a truth function determined by a formula for all possible combinations of inputs. Boolean vectors are just columns of a truth table, which give these outputs in a fixed order. Each such output is an *entry* or *component* of the vector.

In the remaining two sections of this part, which are optional, we use them to demonstrate visually that all truth functions are compositions of the operations of Boolean complementation, multiplication and addition. In the last section, we manipulate vectors to prove the law of duality: if two formulas in which the negation, conjunction and disjunction signs are the only connectives, are truth functionally equivalent, then so are their duals, obtained by substituting $\wedge$'s for $\vee$'s and $\vee$'s for $\wedge$'s throughout.

3.1.1 Boolean Reasoning: Truth Functional Entailment and Validity

God doesn't need to reason. If you already know everything, there is no need to deduce anything. The whole point of demonstrating that the conclusion of an argument logically follows from its assumptions is that you don't *have* to know the truth values of the assumptions in order to know that the argument is valid.

Of course the conclusion of a valid argument contains no more information than the assumptions, but when it's not obvious that the conclusion really follows, it may be interesting, especially if its following has implications for previously unanswered questions. And when the conclusion does not clearly follow, or it's not obvious that it really follows, what you need to find is a demonstration that it follows or that it does not.

Boole's idea was to use logical complementation, multiplication and addition to analyze and evaluate **Boolean reasoning**–those pieces of reasoning which can be done in terms of negation, conjunction and disjunction by the methods of algebra–so that every mistake in Boolean reasoning may be recognized as an "error in calculation".[1]

To be more precise, let a **formal argument** consist of a finite set $X_1, \ldots, X_n$ of **assumptions**, together with a formula Y called the **conclusion**.[2] Then the formal argument $X_1, \ldots, X_n \therefore Y$ is **truth functionally valid**, and we may write $X_1, \ldots, X_n \models Y$ `iff` the conjunction of the assumptions **truth functionally entails** Y, in the sense that $X_1 \wedge \ldots \wedge X_n \Rightarrow Y$ is a tautology, or equivalently, $X_1 \wedge \ldots \wedge X_n \leq Y$ no matter what the truth values of the variables in $X_1 \wedge \ldots \wedge X_n$ and Y.

A **Boolean argument**

$$[X_1], \ldots, [X_k] \therefore [Y] \tag{3.1}$$

consists of a set $[X_1], \ldots, [X_k]$ of Boolean statements, the members of which are called **assumptions**, together with a single Boolean statement $[Y]$ called its **conclusion**, where the assumptions and conclusion have been symbolized (and denote the truth values of the corresponding Boolean statements as shown), by the formulas inside the square brackets.

Then to this Boolean argument there corresponds a formal argument $X_1, \ldots, X_k \therefore Y$. The Boolean argument $[X_1], [X_2], \ldots, [X_k] \therefore [Y]$ is **truth functionally valid** and we may write

$$[X_1], [X_2], \ldots, [X_k] \longrightarrow [Y] \quad \texttt{iff} \quad X_1, X_2, \ldots, X_k \models Y \ ,$$

that is, a Boolean argument is valid `iff` the corresponding formal argument is valid. Thus a Boolean argument is truth functionally valid `iff` no matter what the truth values of the atomic statements which the argument contains, it never happens that all the assumptions are true and the conclusion false. A valid argument with a single assumption therefore entails its conclusion.

---

[1] We also consider Boolean arguments involving material implications. These of course can be defined in terms of negation and either conjunction or disjunction.

But Boolean logic cannot of course provide a complete analysis of mathematical reasoning, for it cannot even handle generalities or statements of existence.

[2] We do not consider arguments with infinitely many assumptions until Chapter 4, where we do so for theoretical purposes. But the problem of determining whether a given argument with infinitely many assumptions is valid is not, in general solvable by calculation, while the corresponding "validity problem" for formal arguments with finitely many assumptions always is, however impractical it may be to carry out the necessary calculations.

C.I. Lewis once defined logic to be the study of the entailment relation. Logic has since outgrown this definition, but the various senses of that concept are still important. In Boolean logic, it reduces to a relation between Boolean *formulas*. In this and the next chapter, we are concerned with Boolean logic, the study of the two-element Boolean algebra, the elements of which are the truth values 0 and 1.

Boolean logic thus includes but is not limited to the study of the truth functional entailment relation between Boolean formulas. For example, truth functional equivalence or mutual entailment is also important, because entailment between *propositions*, which we may (for now) take to be sets of mutually equivalent formulas, lead to the study of algebraic, geometrical and topological structures, the properties of which have important implications for both the power and the limitations of formal logics such as Boolean and first order logics and related systems.

In this chapter, we first show how to determine whether or not an argument is valid, a set of formulas is satisfiable or a pair of formulas is equivalent, and so on, by direct calculation. In the next chapter, we explore other methods, such as formal proofs of arguments, as well as tests of validity and satisfiability using truth trees.

Problems

1. Determine which of the following formulas entail which:

$$\begin{array}{rcl} (A \wedge B) & \vee & (-A \wedge B) \\ A & \Rightarrow & B \\ B & \Rightarrow & A \\ A & \vee & B \end{array}$$

2. Determine whether the following Boolean argument is valid:

   > *This [world] is the best of all possible worlds. This is not the best of all possible worlds. Therefore, God exists.*

   Method: symbolize the two assumptions and the conclusion. You may let B be the truth value of 'this is the best of all possible worlds' and G be the truth value of 'God exists'.

3.1.2 Calculating the Outputs of n-ary Truth Functions for All Possible Inputs Using Truth Tables and Vectors

In the previous chapter, we were concerned with how to calculate the truth value of a formulas, when given the truth values of the variables in it. But in evaluating a Boolean *argument*, to determine whether it's valid, we need to

know the truth values of the formula involved for all possible combinations of values of the variables in their language.

One way of doing this is to calculate the value of the truth functions determined by the formulas, for all possible combinations of inputs. Take, for example, the formula,

$$(A \wedge -B) \vee (-A \wedge B) \ . \tag{3.2}$$

It contains two Boolean variables; ' A ' (i.e. ' A_1 ') and ' B ' (i.e. ' A_2 ') which denote the first and second inputs, respectively. We present the input pairs in a fixed order under ' A ' and ' B ', where the digit under ' A ' denotes the first input, and the digit under ' B ' denotes the second:

A	B
1	1
1	0
0	1
0	0

Here are two ways of remembering that order:

For humanists: if you identify, say, the pair $\langle 1, 0\rangle$ of truth values with 'oz', where 'o' stands for the number one and 'z' stands for the number zero, and you put the letter pairs in alphabetical order *from the top down*, and then replace them by the corresponding truth value pairs, the truth value pairs will be in the correct order.

For computer scientists: if you identify the truth value pair $\langle 1, 0\rangle$ with the *binary numeral* ' 10 ' which denotes $1 \cdot 2 + 0 = 2$, the pairs of truth values corresponding to binary numerals are displayed so that the binary numerals denote consecutive natural numbers starting with 0 *from the bottom up*.

So these two devices for remembering the standard order (for binary truth functions) produce the same order:

letter pair	binary numberal	input pair	A	B
oo	$11 = 1 \cdot 2 + 1 = 3$	$\langle 1, 1\rangle$	1	1
oz	$10 = 1 \cdot 2 + 0 = 3$	$\langle 1, 0\rangle$	1	0
zo	$01 = 0 \cdot 2 + 1 = 3$	$\langle 0, 1\rangle$	0	1
zz	$00 = 0 \cdot 2 + 0 = 3$	$\langle 0, 0\rangle$	0	0

Now to the right of 'A' and 'B', write the subformulas of '$(A \wedge -B) \vee (-A \wedge B)$' which have just one connective. These are ' $-A$ ' and ' $-B$ '. Because ' A ' is a subformula of ' $-A$ ' and ' B ' of ' $-B$ ', the truth values of which can be calculated in just one step, since we were already given the truth values of ' A ' and ' B ':

A	B	$-A$	$-B$
1	1	0	0
1	0	0	1
0	1	1	0
0	0	1	1

Next, on the right, we write the two subformulas of the formula '$(A \wedge -B) \vee (-A \wedge -B)$' which have two connectives, and again we can calculate each of their truth values on every row in one step, multiplying A by $-B$ or $-A$ by B, as the case may be:

A	B	$-A$	$-B$	$A \wedge -B$	$-A \wedge B$
1	1	0	0	0	0
1	0	0	1	1	0
0	1	1	0	0	1
0	0	1	1	0	0

Finally, take the Boolean sum of the entries of each row in the last two columns:

A	B	$-A$	$-B$	$(A \wedge -B)$	$(-A \wedge B)$	$(A \wedge -B) \vee (-A \wedge B)$
1	1	0	0	0	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	0	1	1	0	0	0

Now we have a **truth table** for ' $(A \wedge -B) \vee (-A \wedge B)$ '. It corresponds to the truth function **determined** by the formula, ' $(A \wedge -B) \vee (-A \wedge B)$ ', the internal diagram of which is:

$$\begin{aligned} \langle 1,1\rangle &\longrightarrow 0 \\ \langle 1,0\rangle &\longrightarrow 1 \\ \langle 0,1\rangle &\longrightarrow 1 \\ \langle 0,0\rangle &\longrightarrow 0 \end{aligned}$$

The function determined by ' $(A \wedge -B) \vee (-A \wedge B)$ ' can also be calculated from the matrices of the functions $-$, $\wedge$ and $\vee$, instead of by truth tables. But for the purpose of calculation, truth tables are more efficient, if constructed as we did above.

We call each column of a truth table, such as those of the truth table above, a **Boolean vector**, each column entry being a **component** of the vector. Since the vectors in the table have four components, we call them **4-vectors**. For any number k, two k-vectors are equal `iff` their corresponding components are equal. We write $\mathbf{X}$ or $V(X)$ for the vector for the L_n-formula X. It is a 2^n-vector. When writing the vector in isolation as a "column vector", displaying its components, we enclose the column in braces. For instance, the truth table

above contains seven column vectors. They are:

$$\begin{array}{ccccccc} \mathbf{A} & \mathbf{B} & -\mathbf{A} & -\mathbf{B} & \mathbf{A}\wedge\mathbf{B} & -\mathbf{A}\wedge\mathbf{B} & (\mathbf{A}\wedge-\mathbf{B})\vee(-\mathbf{A}\wedge\mathbf{B}) \\ \left[\begin{array}{c}1\\1\\0\\0\end{array}\right] & \left[\begin{array}{c}1\\0\\1\\0\end{array}\right] & \left[\begin{array}{c}0\\0\\1\\1\end{array}\right] & \left[\begin{array}{c}0\\0\\1\\1\end{array}\right] & \left[\begin{array}{c}0\\1\\0\\0\end{array}\right] & \left[\begin{array}{c}0\\0\\1\\0\end{array}\right] & \left[\begin{array}{c}0\\1\\1\\0\end{array}\right] \end{array}$$

Every 4-vector of L_2 corresponds to a truth function, and *vice versa*. For instance, the internal diagram for the function determined by the truth function f determined by '$(A\wedge -B)\vee(-A\wedge B)$' corresponds to $(\mathbf{A}\wedge-\mathbf{B})\vee(-\mathbf{A}\wedge\mathbf{B})$:

$$\begin{array}{cc} f & (\mathbf{A}\wedge-\mathbf{B})\vee(-\mathbf{A}\wedge\mathbf{B}) \\ \begin{array}{l}\langle 1,1\rangle \longrightarrow 0\\ \langle 1,0\rangle \longrightarrow 1\\ \langle 0,1\rangle \longrightarrow 1\\ \langle 0,0\rangle \longrightarrow 0\end{array} & \left[\begin{array}{c}0\\1\\1\\0\end{array}\right] \end{array} \tag{3.3}$$

Observe too that $-\mathbf{A}$ is obtained by taking the Boolean complement of every component of $\mathbf{A}$, changing every component 1 into a 0 and every 0 into a 1. Similarly $-\mathbf{A}\wedge\mathbf{B}$ is the vector obtained by multiplying every component of $-\mathbf{A}$ by the corresponding component in the same row of $\mathbf{B}$, and that to obtain $(\mathbf{A}\wedge-\mathbf{B})\vee(-\mathbf{A}\wedge\mathbf{B})$, just take the Boolean sum of each component of $\mathbf{A}\wedge-\mathbf{B}$ and corresponding component of $-\mathbf{A}\wedge\mathbf{B}$. Thus if X and Y are formulas,

$$\begin{aligned} V(-X) &= -V(X)\ , \\ V(X\star Y) &= V(X)\star V(Y)\ , \end{aligned}$$

where $\star$ is a binary connective.

Two k-vectors are equal `iff` their corresponding components are equal. We also define $\mathbf{X}\leq\mathbf{Y}$ `iff` $\mathbf{X}$ and $\mathbf{Y}$ are n-vectors, and each component x_i of X $(1\leq i\leq n)$ is less than or equal to the corresponding component y_i of $\mathbf{Y}$: $x_i\leq y_i$. Moreover, if $\mathbf{X}\leq\mathbf{Y}$ and $\mathbf{Y}\leq\mathbf{X}$, then $\mathbf{X}=\mathbf{Y}$, since both $x_i\leq y_i$ and $y_i\leq x_i$ for each component x_i and the corresponding component y_i. It is then immediate that $x_i=y_i$, from which it follows that $\mathbf{X}=\mathbf{Y}$, since all corresponding components of each vector are equal. For instance, we see by inspection that:

$$\begin{array}{ccccc} (-\mathbf{A}\vee\mathbf{B})\wedge(\mathbf{A}\vee-\mathbf{B}) & = & \mathbf{A}\Leftrightarrow\mathbf{B} & \leq & \mathbf{A}\Rightarrow\mathbf{B} \\ \left[\begin{array}{c}1\\0\\0\\1\end{array}\right] & = & \left[\begin{array}{c}1\\0\\0\\1\end{array}\right] & \leq & \left[\begin{array}{c}1\\0\\0\\1\end{array}\right] \end{array} \tag{3.4}$$

In general, then, $X\models Y$ `iff` $\mathbf{X}\leq\mathbf{Y}$, and $X\equiv Y$ `iff` $\mathbf{X}=\mathbf{Y}$. That is, X entails Y `iff` the vector for X is less than or equal to the vector for Y, and X and Y are truth functionally equivalent `iff` they have the same vectors.

Another example: since $-\mathbf{A} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ and $-\mathbf{B} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$, we have

$$-(\mathbf{A} \wedge \mathbf{B}) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = -\mathbf{A} \vee -\mathbf{B} \; ; \qquad -(\mathbf{A} \vee \mathbf{B}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = -\mathbf{A} \wedge -\mathbf{B}$$

which establish the following equivalences, known as **DeMorgan's Laws**:

$$\begin{aligned} -(A \wedge B) &\equiv -A \vee -B \\ -(A \vee B) &\equiv -A \wedge -B \end{aligned}$$

Problems

1. (a) Using DeMorgan's Laws, show that ' $\vee$ ' can be defined in terms of ' $-$ ' and ' $\wedge$ ', i.e. find a formula truth functionally equivalent to ' $A \vee B$ ' containing no Boolean constants, and in which ' $-$ ' and ' $\wedge$ ' are the only connectives to occur.

   (b) Show that ' $\wedge$ ' can be defined in terms of ' $-$ ' and ' $\vee$ '.

2. (a) Show that ' $\Rightarrow$ ' can be defined in terms of ' $-$ ' and ' $\vee$ '. (Hint: first show that $\mathbf{A} \Rightarrow \mathbf{B} = -\mathbf{A} \vee \mathbf{B}$).

   (b) Show that ' $\Rightarrow$ ' can be defined in terms of ' $-$ ' and ' $\wedge$ '. (Hint: find a formula equivalent to ' $A \Rightarrow B$ ' where ' $-$ ' and ' $\wedge$ ' are the only connectives to appear, and using vectors or truth tables, show that it is indeed equivalent to ' $A \Rightarrow B$ '.)

3. Give a truth table for '$(B \wedge C) \wedge ((E \vee R) \wedge -(E \wedge R))$'. How many rows does it have?

3.1.3 Truth Value Assignments, Boolean Valuations and Validity

While each L_n-formula X determines an n-ary truth function once the truth values of its n inputs $A_1, \ldots, A_n$ are given, a **truth value assignment** $\sigma : Af \longrightarrow \{0,1\}$ assigns a truth value to each atomic formula of L for it to denote (subject to the condition that '0' must always denote 0 and '1' must always denote 1 if L_n has either constant). Thus a **truth value assignment** (`TVA`) is a function which assigns to each variable A_i in Bv_n a truth value A_i, and is just another way of specifying what the particular inputs $A_1, \ldots, A_n$ are.

So we introduce a new point of view, which provides a useful and common method for defining validity and related concepts, which is all but essential in infinitely generated languages, such as L_∞ which we need at the end of Chapter 4, and zero order and first order languages which will be our main concern from

Chapter 6 on, where both the rows and columns of a full truth table could have infinitely many entries.

Take, for example a truth value assignment σ of a finitely generated language L_2 which lacks constants, where σ has the internal diagram:

$$\sigma(\text{`}A\text{'}) \longrightarrow 1 , \tag{3.5}$$
$$\sigma(\text{`}B\text{'}) \longrightarrow 0 . \tag{3.6}$$

In section 2.2.2 (page 33), for instance where we now write $\sigma(\text{`}A\text{'}) = 1$, we would have written $A = 1$, and we would have written $B = 0$ instead of $\sigma(\text{`}B\text{'}) = 0$. In either case, whether the inputs of the function f determined by the formula F are given by equations of either form, the way in which we find find the truth value $\tau(F)$ that F denotes is basically the same.

So, '0' denote 0, '1' denote 1, and if R is a Boolean variable, $R = 1$ or $R = 0$, depending on whether $\sigma(R) = \tau(R)$ is 1 or 0, as the case may be. Moreover,

$$\tau(-X) = -\tau(X) , \tag{3.7}$$

that is, $-X$ denotes the Boolean complement of what X denotes [which is $\tau(X)$],

$$\tau(X \wedge Y) = \tau(X) \wedge \tau(Y) , \tag{3.8}$$

($X \wedge Y$ denotes the logical product of what X and Y denote),

$$\tau(X \vee Y) = \tau(X) \vee \tau(Y) , \tag{3.9}$$

($X \vee Y$ denotes the logical sum of what X and Y denote), and

$$\tau(X \Rightarrow Y) = \tau(X) \Rightarrow \tau(Y) . \tag{3.10}$$

Thus each truth value assignment $\sigma : Af \longrightarrow \{0, 1\}$ of a Boolean language L the set of atomic formulas of which is Af, such that $\sigma(\text{`}0\text{'}) = 0$ and $\sigma(\text{`}1\text{'}) = 1$ (if L has the constants ' 0 ' or ' 1 '), determines a unique **Boolean valuation** $\tau : F(Af) \longrightarrow \{0, 1\}$ which extends σ, subject to these conditions:

$$\begin{aligned} \sigma(T) &= \tau(T) \quad (\text{if } T \text{ is atomic}) \\ \tau(-X) &= -\tau(X) \\ \tau(X \star Y) &= \tau(X) \star \tau(Y) \quad (\text{if } \star \text{ is a binary connective}) \end{aligned}$$

If, say F is the formula ' $-A \wedge (A \vee B)$ ', the three ways of calculating $\tau(F)$ are:

Method of section 2.2.2 (page 33): Given that $A = 1$ and $B = 0$,

$$\begin{aligned} -A \wedge (A \vee B) &= -1 \wedge (1 \vee 0) \\ &= 0 \wedge 1 \\ &= 0 \end{aligned}$$

Method of section 3.1.2 (page 67): Given that f determines the formula: ‘ $-A \wedge (A \vee B)$ ’,

$$\begin{aligned} f(0,1) &= -1 \wedge (1 \vee 0) \\ &= 0 \wedge 1 \\ &= 0 \end{aligned}$$

Method of `TVA` (page 71): Given that $\sigma(\text{‘ } A \text{ ’}) = 1$ and $\sigma(\text{‘ } B \text{ ’}) = 0$,

$$\begin{aligned} \tau(\text{‘ } -A \wedge (A \vee B) \text{ ’}) &= -\sigma(\text{‘ } A \text{ ’}) \wedge \big(\sigma(\text{‘ } A \text{ ’}) \vee \sigma(\text{‘ } B \text{ ’})\big) \\ &= -1 \wedge (1 \vee 0) \\ &= 0 \wedge 1 \\ &= 0 \end{aligned}$$

All this was presupposed in section 2.2.2 (page 33) and section 3.1.1 (page 65), where we substituted the Boolean constants, ‘ 0 ’ or ‘ 1 ’ as the case may be, for each variable θ in the formula X, according to whether $\sigma(\theta) = 0$ or $\sigma(\theta) = 1$, which allowed us to calculate the truth value of X (given σ). There are no restrictions on what σ assigns to a variable: you are free to assign any truth value you like to a variable in X, *independently* of what you assign to any other variables. But of course it is always the case that $\sigma(\text{‘ } 0 \text{ ’}) = 0$ and $\sigma(\text{‘ } 1 \text{ ’}) = 1$.

The truth value assignment σ **satisfies** X `iff` $\tau(X) = 1$, where τ is the Boolean valuation corresponding to σ, in which case τ also satisfies X. And σ (or the corresponding valuation τ) **satisfies** the set $X_1, \ldots, X_n$ of L-formulas `iff` it satisfies every formula in the set.

Then we may define $X_1, \ldots, X_n \models Y$ to hold `iff` every valuation which satisfies $X_1, \ldots, X_n$ satisfies Y, or equivalently, the set $X_1, \ldots, X_n, -Y$ is **unsatisfiable**, that is, no valuation satisfies it. So if $X_1, \ldots, X_n, -Y$ is **satisfiable**, then some truth value assignment satisfies it, and any such function σ is a **counterexample** to the claim that $X_1, \ldots, X_n \models Y$ holds. A **valid** argument is therefore an argument that has no counterexamples. Observe too that every row of a truth table for a formula F determines a truth value assignment.

In practice, to specify a valuation, it is easiest to provide the corresponding truth value assignment, either by means of a set of equations, an internal diagram or a one row truth table. For instance, here are four ways you can provide a counterexample to the claim that the set $-A$, $A \Rightarrow -B$ is satisfiable:

Equations
$A = 0$
$B = 0$

Internal Diagram
‘ A ’ $\longrightarrow 0$
‘ B ’ $\longrightarrow 0$

One Row Table	
A	B
0	0

Full one row truth table

A	B	$-A$	$-B$	$A \Rightarrow -B$
0	0	1	1	1

The equation set $A = 0$, $B = 0$ is a **solution** to the simultaneous equations $-A = 1$, $A \Rightarrow -B = 1$, for $-0 = 1$ and $0 \Rightarrow -0 = 1$.

Problems

1. Find a counterexample to the argument: $A \Rightarrow B \therefore B \Rightarrow A$.

3.1.4 Basis Vectors*

We can use vectors whenever we are concerned with a finitely generated Boolean language L_n. In this section and the next, we will be working within L_2, the variables of which are ' A ' and ' B ', (i.e. ' A_1 ' and ' A_2 '). We assume as usual that ' A ' denotes the first input and ' B ' the second input of the binary truth functions determined by L_2-formulas, and so we restrict our attention to 4-vectors only.

If U is an arbitrary 4-vector with components u_1, u_2, u_3 and u_4, so that,

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \tag{3.11}$$

then U is the column of outputs of a binary truth function $f : \{0,1\}^2 \longrightarrow \{0,1\}$ with the internal diagram,

$$\begin{array}{lcl} \langle 1,1\rangle & \longrightarrow & u_1 \\ \langle 1,0\rangle & \longrightarrow & u_2 \\ \langle 0,1\rangle & \longrightarrow & u_3 \\ \langle 0,0\rangle & \longrightarrow & u_4 \end{array}$$

where the input pairs are listed in the same standard order as the corresponding first two entries in the truth table for any formula F that determines f:

A	B	F
1	1	u_1
1	0	u_2
0	1	u_3
0	0	u_5

There are 16 4-vectors, since u_1 can be either 0 or 1, and so can u_2, u_3 and u_4, making 16 possibilities in all. The vectors **A** and **B** are of course,

$$\mathbf{A} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

*This part may be skipped without loss of continuity.

and their components are the first and second inputs of f, respectively. Then $\mathbf{A}$ and $\mathbf{B}$ correspond to the functions $\langle A, B\rangle \longmapsto A$ and $\langle A, B\rangle \longmapsto B$ determined by the formulas ' A ' and ' B ' with the internal diagrams,

$\langle A, B\rangle \longmapsto A$	$\langle A, B\rangle \longmapsto B$
$\langle 1, 1\rangle \longmapsto 1$	$\langle 1, 1\rangle \longmapsto 1$
$\langle 1, 0\rangle \longmapsto 1$	$\langle 1, 0\rangle \longmapsto 0$
$\langle 0, 1\rangle \longmapsto 0$	$\langle 0, 1\rangle \longmapsto 1$
$\langle 0, 0\rangle \longmapsto 0$	$\langle 0, 0\rangle \longmapsto 0$

and are the two **projection functions** on $\{0, 1\}^2$. We also have the **unit 4-vector 1**, all of whose components are 1, and the **zero vector 0**, all of whose components are 0, and they correspond to the two constant functions on $\{0, 1\}^2$.

There are also four **basis vectors**, which are vectors having a single entry which is 1, the other entries being 0. For convenience, we call these e_1, e_2, e_3 and e_4, where e_1 has 1 for the top entry, the other entries being 0, e_2 has 1 as its second entry from the top, the other entries being 0, and so on.[3]

It is evident that every vector except the zero vector is either a basis vector or the logical sum of basis vectors. For example, let $\mathbf{A} = e_1 \vee e_2$, and $\mathbf{B} = e_1 \vee e_3$, then $\mathbf{A} \wedge \mathbf{B} = e_1$, $\mathbf{A} \vee \mathbf{B} = e_1 \vee e_2 \vee e_3$, and $\mathbf{A} \Rightarrow \mathbf{B} = e_1 \vee e_3 \vee e_4$:

$$\mathbf{A} \quad = \quad e_1 \quad \vee \quad e_2 \qquad\qquad \mathbf{B} \quad = \quad e_1 \quad \vee \quad e_3$$

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{A} \quad \wedge \quad \mathbf{B} \quad = \quad \mathbf{A} \wedge \mathbf{B} \quad = \quad e_1$$

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \wedge 1 \\ 1 \wedge 0 \\ 0 \wedge 1 \\ 0 \wedge 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$e_1 \quad \vee \quad e_2 \quad \vee \quad e_3 \quad = \quad \mathbf{A} \vee \mathbf{B}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

[3]The vectors e_1, e_2, e_3 and e_4 may be taken to be the basis vectors for a four dimensional vector space in linear algebra, where they are unit vectors, since they have length 1. But as we do not define lengths for Boolean vectors, no confusion should arise in defining Boolean vectors to be unit vectors when all of their entries are 1. However, the set of Boolean vectors of L_2 does form a four dimensional projective space. See Menger (1936)[60].

$$\mathbf{A} \Rightarrow \mathbf{B} \quad = \quad e_1 \quad \vee \quad e_3 \quad \vee \quad e_4 \quad = \quad \mathbf{A} \Rightarrow \mathbf{B}$$

$$\begin{bmatrix} 1 \Rightarrow 1 \\ 1 \Rightarrow 0 \\ 0 \Rightarrow 1 \\ 0 \Rightarrow 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

The unit vector is the logical sum of all the basis vectors, and since it is the vector of any tautology, it is also $\mathbf{A} \vee -\mathbf{A}$. And since the zero vector is the vector for any (truth-functional) contradiction, it is $\mathbf{A} \wedge -\mathbf{A}$.

It is also clear that every basis vector of a language with n Boolean variables is the vector for a **complete clause**, in which each of the n variables occurs exactly once, and which is a conjunction of the form $R_1 \wedge \ldots \wedge R_n$, each conjunct R_i of which is a **literal**, either ' A_i ' or ' $-A_i$ '. Since there are two Boolean variables of L_2, the language we are considering, L_2 has four complete clauses, and each of its basis vectors is the vector of a complete clause.

So we have,

$$\begin{aligned} e_1 &= \mathbf{A} \wedge \mathbf{B} , \\ e_2 &= \mathbf{A} \wedge -\mathbf{B} , \\ e_3 &= -\mathbf{A} \wedge \mathbf{B} , \\ e_4 &= -\mathbf{A} \wedge -\mathbf{B} . \end{aligned}$$

There is thus a one-to-one correspondence between basis vectors and complete clauses and between complete clauses and truth value assignments. That is, every basis vector is the vector for exactly one complete clause, which in turn is satisfied by one and only one `TVA` (page 71).

And since the zero vector is $\mathbf{A} \wedge -\mathbf{A}$, while every non- zero vector is a basis vector or the logical sum of basis vectors, it follows that every formula which is not a contradiction is equivalent to a complete clause, or to a disjunction of complete clauses. Therefore, *Every L_2-formula is equivalent to a formula in which the only connectives to occur are '$-$ ', '$\wedge$ ' and '$\vee$ '.*

Problems

1. (a) Write the vectors $\mathbf{A} \vee \mathbf{B}$ and $\mathbf{A} + \mathbf{B}$ as the logical sums of L_2-basis vectors.

   (b) Write the formulas ' $A \vee B$ ' and ' $A + B$ ' as the disjunctions of complete clauses of L_2.

3.1.5 Vectors and Duality*

Let L be an L_2 language in which ' $-$ ', ' $\wedge$ ' and ' $\vee$ ' are the only connectives to occur. The **dual** $\star X$ of a formula X in which ' $\wedge$ ',' $\vee$ ' and ' $-$ ' are the only connectives to occur, results from X by exchanging $\wedge$'s and $\vee$'s in X throughout and also $\top$'s (or 1's) and $\bot$'s (or 0's) if any. For instance, the dual of ' $A \vee 0$ ' is ' $A \wedge 1$ ', and *vice versa*, and ' $A \wedge (A \vee B)$ ' and ' $A \vee (A \wedge B)$ ' are dual to each other. Take the truth table for ' $A \wedge B$ '. Each entry under $A \wedge B$ is the logical *product* of the two entries to the left:

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

(3.12)

Now negate all entries, exchanging 1's and 0's throughout. This changes $\mathbf{A}$ to $-\mathbf{A}$ and $\mathbf{B}$ to $-\mathbf{B}$ and $\mathbf{A} \wedge \mathbf{B}$ to $-(\mathbf{A} \wedge \mathbf{B})$:

$-A$	$-B$	$-(A \wedge B)$
0	0	0
0	1	1
1	0	1
1	1	1

(3.13)

Then each entry in the right hand column is the logical *sum* of the corresponding entries in the first two, that is, $-(\mathbf{A} \wedge \mathbf{B}) = -\mathbf{A} \vee -\mathbf{B}$. Now if you turn *all* the columns of the table (3.13) upside down, each one is the **inversion** ¡$\mathbf{X}$ of the corresponding vector $\mathbf{X}$ in the table (3.13), and what you get is the truth table for ' $A \vee B$ ', which is the dual ' $\star(A \wedge B)$ ' of ' $A \wedge B$ ':

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

Thus we define the **dual** $\star\mathbf{X}$ of any vector $\mathbf{X}$ to be $-$¡$\mathbf{X}$.

Observe that,

$$\begin{aligned} -(¡¡)\mathbf{X} &= (-¡)¡\mathbf{X} \,, \\ -¡\mathbf{X} &= ¡-\mathbf{X} \,, \\ ¡¡\mathbf{X} &= \mathbf{X} \,, \\ ¡\mathbf{A} &= -\mathbf{A} \,, \\ ¡\mathbf{B} &= -\mathbf{B} \,, \end{aligned} \tag{3.14}$$

*This part may be skipped without loss of continuity.

$$\begin{aligned} ¡(\mathbf{A} \wedge \mathbf{B}) &= ¡\mathbf{A} \wedge ¡\mathbf{B} \ , \\ ¡(\mathbf{A} \vee \mathbf{B}) &= ¡\mathbf{A} \vee ¡\mathbf{B} \ . \end{aligned}$$

It thus follows that,

$$\begin{aligned} ¡(\mathbf{A} \wedge \mathbf{B}) &= -\mathbf{A} \wedge -\mathbf{B} \ , \\ ¡(\mathbf{A} \vee \mathbf{B}) &= -\mathbf{A} \vee -\mathbf{B} \ , \end{aligned}$$

and we have:

$$¡\mathbf{A} \wedge ¡\mathbf{B} = -\mathbf{A} \wedge -\mathbf{B} \qquad ¡\mathbf{A} \vee ¡\mathbf{B} = -\mathbf{A} \vee -\mathbf{B}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \wedge \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \ , \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Also observe:

$$\star\mathbf{A} \vee \star\mathbf{B} = \star(\mathbf{A} \wedge \mathbf{B}) \qquad \star\mathbf{A} \wedge \star\mathbf{B} = \star(\mathbf{A} \vee \mathbf{B})$$

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \vee \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \ , \qquad \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Given equations (3.14), it follows from DeMorgan's laws that,

$$\begin{aligned} \star(\mathbf{A} \wedge \mathbf{B}) &= \mathbf{A} \vee \mathbf{B} \ , \\ \star(\mathbf{A} \vee \mathbf{B}) &= \mathbf{A} \wedge \mathbf{B} \ , \end{aligned} \tag{3.15}$$

for we have the following identities,

$$\begin{aligned} \star(\mathbf{A} \wedge \mathbf{B}) &= -¡(\mathbf{A} \wedge \mathbf{B}) \\ &= -(¡\mathbf{A} \wedge \mathbf{B}) \\ &= -¡\mathbf{A} \vee -¡\mathbf{B} \\ &= --\mathbf{A} \vee --\mathbf{B}) \\ &= \mathbf{A} \vee \mathbf{B} \ . \end{aligned} \tag{3.16}$$

CRAIG WERE THESE EQUATIONS TO BE DELETED?

$$\begin{aligned} -¡(\mathbf{A} \wedge \mathbf{B}) = -(¡\mathbf{A} \wedge ¡\mathbf{B}) &= -¡\mathbf{A} \wedge -¡\mathbf{B} = --\mathbf{A} \vee --\mathbf{B} = \mathbf{A} \vee \mathbf{B} \\ ¡ - (\mathbf{A} \wedge \mathbf{B}) = ¡(-\mathbf{A} \vee -\mathbf{B}) &= ¡ - \mathbf{A} \vee ¡ - \mathbf{B} = ¡¡\mathbf{A} \vee ¡¡\mathbf{B} = \mathbf{A} \vee \mathbf{B} \end{aligned}$$

Dually,

$$\star(\mathbf{A} \vee \mathbf{B}) = (\mathbf{A} \wedge \mathbf{B}) \ . \tag{3.17}$$

Conversely, given (3.14) and (3.15), DeMorgan's laws follow, for we have the following identities:

$$-(\mathbf{A} \wedge \mathbf{B}) \quad = \quad -(\text{¡¡})(\mathbf{A} \wedge \mathbf{B}) \; . \tag{3.18}$$

WAS THERE SUPPOSED TO BE A SECOND IDENTITY HERE?

And given that $-(\text{¡¡})\mathbf{X} = (-\text{¡})\text{¡}\mathbf{X}$, DeMorgan's laws follow:

$$\begin{aligned} -(\mathbf{A} \wedge \mathbf{B}) = -\text{¡¡}\mathbf{A} \wedge \mathbf{B} \quad &= \quad x \star \text{¡}\mathbf{A} \wedge \mathbf{B} = \star(\text{¡}\mathbf{A} \wedge \text{¡}\mathbf{B}) = \star(-\mathbf{A} \wedge -\mathbf{B}) \\ -(\mathbf{A} \vee \mathbf{B}) = -\text{¡¡}\mathbf{A} \vee \mathbf{B} \quad &= \quad x \star \text{¡}\mathbf{A} \vee \mathbf{B} = \star(\text{¡}\mathbf{A} \vee \text{¡}\mathbf{B}) = \star(-\mathbf{A} \vee -\mathbf{B}) \end{aligned}$$

It is now clear that for any formulas X and Y, we have,

$$\begin{aligned} \text{¡}(\mathbf{X} \wedge \mathbf{Y}) \quad &= \quad \text{¡}\mathbf{X} \wedge \text{¡}\mathbf{Y} \; , \\ \text{¡}(\mathbf{X} \vee \mathbf{Y}) \quad &= \quad \text{¡}\mathbf{X} \vee \text{¡}\mathbf{Y} \; . \end{aligned}$$

So by DeMorgan's laws,

$$\begin{aligned} \star(\mathbf{X} \wedge \mathbf{Y}) = -\text{¡}(\mathbf{X} \wedge \mathbf{Y}) \quad &= \quad -(\text{¡}\mathbf{X} \wedge \text{¡}\mathbf{Y}) = (-\text{¡}\mathbf{X} \vee -\text{¡}\mathbf{Y}) = (\star\mathbf{X} \vee \star\mathbf{Y}) \; , \\ \star(\mathbf{X} \vee \mathbf{Y}) = -\text{¡}(\mathbf{X} \vee \mathbf{Y}) \quad &= \quad -(\text{¡}\mathbf{X} \vee \text{¡}\mathbf{Y}) = (-\text{¡}\mathbf{X} \wedge -\text{¡}\mathbf{Y}) = (\star\mathbf{X} \wedge \star\mathbf{Y}) \; , \end{aligned}$$

If X is a formula of any L_2 language in which ' $-$ ', ' $\wedge$ ' and ' $\vee$ ' are the only connectives, we see that $V(\star Z) = \star V(Z)$. For instance, we have,

$$\begin{aligned} \star V\Big((A \wedge B) \vee A\Big) \quad &= \quad \star V(A \wedge B) \vee \star V(A) \; , \\ &= \quad \Big(V(\star A) \vee \star V(B)\Big) \wedge V(\star A) \; , \\ &= \quad \Big(V(A) \vee V(B)\Big) \wedge V(A) \; , \\ &= \quad V(A \vee B) \wedge V(A) \; , \\ &= \quad V\Big((A \vee B) \wedge A\Big) \; , \\ &= \quad V\Big(\star((A \wedge B) \vee A)\Big) \; . \end{aligned}$$

The **Law of Duality** now follows:

> `If two formulas are equivalent, so are their duals. That is, if` $X \equiv Y$`, then` $\star X \equiv \star Y$`.`

For if $X \equiv Y$, then $V(X) = V(Y)$, so that $\star V(X) = \star V(Y)$, and consequently, $V(\star X) = V(\star Y)$, and thus $\star X \equiv \star Y$.

Problems

1. If $\mathbf{R}$ is the 4-vector for a literal of L_2, either $\mathbf{A}$, $-\mathbf{A}$, $\mathbf{B}$, or $-\mathbf{B}$, then the inversion of $\mathbf{R}$ is its complement, so that the inversion of the complement of $\mathbf{R}$ is $\mathbf{R}$ itself, and $\mathbf{R}$ is its own dual. Is there any 4-vector, the inversion of which is its dual? Why or why not?

2. Let $e^1 = \mathbf{A} \vee \mathbf{B}$, $e^2 = \mathbf{A} \vee -\mathbf{B}$, $e^3 = -\mathbf{A} \vee \mathbf{B}$ and $e^4 = -\mathbf{A} \vee -\mathbf{B}$. Then e^1, e^2, e^3 and e^4 are the **dual basis vectors** of L_2. Show that the vector for every non-tautologous L_2-formula is either a dual basis vector or the logical product of dual basis vectors.

3. Explain why $-(¡¡)\mathbf{X} = (-¡)-\mathbf{X}$. [Hint: observe that the unary functions $-$ and ¡ are permutations, and compositions of permutations of the same set are always defined, and that compositions are always associative whenever they are defined (as remarked in Chapter 1)].

3.2 The Basic Problems of Boolean Logic

The **satisfiability problem** is the problem of determining whether a given set of formulas is *satisfiable*, that is, whether they can all be true at once. The **validity problem** is the problem of determining whether a given argument is valid, and the *equivalence problem* is the problem of determining whether two formulas are truth functionally equivalent.

Since truth tables for formulas of a Boolean language L determine a Boolean valuation of L, you can determine by truth table whether a given set $X_1, \ldots, X_k$ is satisfiable. Just determine whether there is a row of the truth table for the conjunction $X_1 \wedge \ldots \wedge X_k$ in which the conjunction is true. That row determines a truth value assignment and so also a valuation of L which *satisfies* $X_1 \wedge \ldots \wedge X_k$. If there is such a row, then the set $X_1, \ldots, X_k$ is indeed satisfiable.

The validity problem can now be solved: the argument $X_1 \wedge \ldots \wedge X_k \therefore Y$ is *valid* `iff` $X_1, \ldots, X_k, \ -Y$ is *unsatisfiable*, i.e. no valuation of L satisfies this set. Of course X entails Y `iff` $X, -Y$ is unsatisfiable, and equivalence is mutual entailment: $X \equiv Y$ `iff` both $X \models Y$ and $Y \models X$.

3.2.1 The Basic Problems of Boolean Logic, and How to Solve Them by Calculations with Truth Tables or Vectors

The problems in Boolean logic most often discussed are: the validity problem, the satisfiability problem, the entailment problem, the tautology problem and the equivalence problem. Two of these, the satisfiability problem and the validity problem involve sets of formulas.

In finitely generated Boolean languages any set of formulas, whether finite or infinite, contain only finitely many non-equivalent formulas. In L_1, for example, since the vector for any formula is one of four 2-vectors, any set of non-equivalent L_1-formulas has at most four members, and because every L_2-formula has one of sixteen 4-vectors, a set of non-equivalent L_2-formulas can have no more than sixteen members.

We restrict our attention to those cases of the basic problems which involve finite sets of formulas only. This has the advantage that they can all be solved by truth tables, although in all but the simplest cases, it is impractical to do so,

for the number of rows of a truth table doubles with each new variable: there are four 2-vectors, sixteen 4-vectors, 256 8-vectors, and in general, 2^{2^n} vectors with 2^n entries.

For a negative solution by truth table to the satisfiability problem, you need to find the vectors for all the formulas concerned, and the same holds for a positive solution to the other four problems. But for a positive solution to the satisfiability problem or a negative solution to the other four problems, all you need is a counterexample, and usually but not always, you will find a row of the truth table which provides one before completing the entire table. There is no known way of finding counterexamples which is shorter than the truth table method all of the time. All you need is luck.

1. The Satisfiability Problem

To determine whether the formula X or the set $S = \{X_1, X_2, \ldots, X_k\}$ of formulas is satisfiable:

> X is satisfiable `iff` $\mathbf{X}$ is not the zero vector. Any non-zero entry in $\mathbf{X}$ corresponds to a truth value assignment (and determines a valuation), which provides an **example** which shows that X is satisfiable, or a **counter-example** to the claim that X is unsatisfiable. S is satisfiable `iff` the conjunction,
>
> $$X_1 \wedge X_2 \wedge \ldots \wedge X_k$$
>
> is satisfiable. Thus S is satisfiable if the logical product of vectors,
>
> $$\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \ldots \wedge \mathbf{X}_k$$
>
> is not the zero vector. Otherwise, S is unsatisfiable. Any non-zero entry in the logical product vector corresponds to a truth value assignment (and determines a valuation), providing an **example** which shows that S is satisfiable, or a **counter-example** to the claim that S is unsatisfiable.

2. The Tautology Problem

To determine whether the formula X is a tautology:

> If $\mathbf{X}$ is the unit vector, then X is a tautology; otherwise not. Equivalently, X is a tautology `iff` $-X$ is unsatisfiable.

3. The Validity Problem

To determine whether $X_1, X_2, \ldots, X_k \models Y$:

> $X_1, X_2, \ldots, X_k \models Y$ `iff` $X_1 \wedge X_2 \wedge \ldots \wedge X_k$ semantically or truth-functionally entails Y, which is true `iff`
>
> $$\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \ldots \wedge \mathbf{X}_k \leq \mathbf{Y} \ .$$

Any truth value assignment which satisfies the set $\{X_1, X_2, \ldots, X_k, Y\}$ is a **counter-example** to the claim that $X_1, X_2, \ldots, X_k \models Y$.

4. The Entailment Problem

To determine whether X truth functionally entails Y:

This is a special case of the validity problem, since by definition, X truth functionally entails Y `iff` $X \models Y$.

5. The Equivalence Problem

To determine whether X and Y are truth functionally equivalent:

X and Y are truth functionally equivalent `iff` they entail each other, so that both $\mathbf{X} \leq \mathbf{Y}$ and $\mathbf{Y} \leq \mathbf{X}$. Thus $X \equiv Y$ `iff` $\mathbf{X} = \mathbf{Y}$. Any pair of differing corresponding entries x_i, y_i of the vectors $\mathbf{X}$ and $\mathbf{Y}$ with $x_i \neq y_i$ yields a counter-example to the claim that X and Y are equivalent.

As we have just seen, while showing how to solve the problems above, all five of them are equivalent, in that if you can solve any one of them, you can solve them all, for:

- If you can solve the satisfiability problem, you can solve the tautology problem. (Because $\models X$ `iff` $-X$is unsatisfiable).
- If you can solve the tautology problem, you can solve the validity problem. (For $X_1, X_2, \ldots, X_k \models Y$ `iff` $\models X_1 \wedge X_2 \wedge \ldots \wedge X_k \Rightarrow Y$).
- If you can solve the validity problem, you can solve the entailment problem. (For X entails Y `iff` $X \models Y$).
- If you can solve the entailment problem, you can solve equivalence problem (Because $X \equiv Y$ `iff` $X \models Y$ and $Y \models X$).
- If you can solve the equivalence problem, you can solve the satisfiability problem. (For $\{X_1, X_2, \ldots, X_k\}$ is unsatisfiable `iff` $X_1 \wedge X_2 \wedge \ldots \wedge X_k \equiv 0$).

Of course, you can always solve word problems involving Boolean statements only, by symbolizing the Boolean statements and then solving the corresponding formal problem. For instance, the proposition that a map consisting of California, Nevada and Utah can be colored by only two colors (say yellow and red) is equivalent to demonstrating the satisfiability of the following set of formulas (where + is algebraic addition):

$$C \quad + \quad A \,,$$

$$\begin{aligned} N &+ V\,, \\ U &+ T\,, \\ -(C &\wedge N)\,, \\ -(N &\wedge U)\,, \\ -(A &\wedge V)\,, \\ -(V &\wedge T)\,. \end{aligned}$$

Here we have the following symbolization:

Boolean Statement	*Truth Value*
`CA` is colored yellow	C
`CA` is colored red	A
`NV` is colored yellow	N
`NV` is colored red	V
`UT` is colored yellow	U
`UT` is colored red	T

To solve this problem by means vectors requires vectors with 64 entries, obtained from 64-row truth tables. In this particular case we do have a short cut: it is easier to draw a map and then supply the proper truth value assignment to prove satisfiability. But again, as far as we know, we're not always so lucky. For instance, it is known that the three color map problem is a "difficult" problem (if the satisfiability problem is one) in the sense that in general, there is no method of solving it which is invariably faster than truth tables, which double in size with the addition of every new variable. Solving the problem for four countries or states as in the problem below would require 4,096 rows, while for eight countries you would need 16,777,216 rows.

Problems

1. Show that it is possible to color a map consisting of California, Oregon, Nevada and Arizona with three colors, but not with two.

2. Show that the Boolean statement *"if Pluto is a Kuiper belt object if it's not a Kuiper belt object, then it is indeed a Kuiper belt object"* cannot be false.

3.3 More on Boolean Syntax and Semantics*

Boolean equations can be used to solve the basic problems of Boolean logic, and related word problems, much as algebraic equations can be used to solve analogous problems in ordinary algebra. The methods we use to solve them allow short cuts which often result in fewer steps than would be required by truth tables, and are less tedious as well.

*This part may be skipped without loss of continuity.

Since Boolean variables can only denote one of two numbers, Boolean equations are in one sense easier to solve than general simultaneous equations in ordinary algebra, but the number of steps required to solve simultaneous Boolean equations in $k+1$ variables is in general twice the number required to solve Boolean equations in k variables. The problem of solving them is therefore "intractable", as far as is known, or impractical for applications involving significantly more variables than those appearing in exercises in logic books.

3.3.1 Boolean Equations

In ordinary algebra, equations can be used to solve word problems. In Boolean logic, the use of Boolean equations provides a natural method by which to solve the satisfiability problem, and so also the other basic problems of Boolean logic. For X is satisfiable `iff` the equation $X = 1$ has a solution, and the set $X_1, X_2, \ldots X_n$ is satisfiable `iff` the set $X_1 = 1, \ldots X_n = 1$ of simultaneous equations has a solution, which is so `iff` the equation $X_1 \wedge \ldots \wedge X_n = 1$ has a solution.

Observe also that $X_1, X_2, \ldots X_n \models Y$ `iff` the set of equations,

$$X_1 = 1, \ldots X_n = 1, Y = 0 \ , \tag{3.19}$$

has no solution, or equivalently the equation $X_1 \wedge \ldots \wedge X_n \wedge -Y = 1$ has no solution. Any solution to the system of simultaneous equations eq. (3.19) provides a **counterexample** to the claim that $X_1, \ldots, X_n \models Y$.

So we are interested in finding solutions to sets of Boolean equations of the form $X = 1$ or $X = 0$, where X is a formula of a Boolean language L, or in showing that a particular system of equations of this form has no solution[4]. This involves no loss of generality, since all Boolean equations can be put in either form, for $X = Y$ `iff` $X \Leftrightarrow Y = 1$, and also $X = Y$ `iff` $X + Y = 0$, and finally, $X = 0$ `iff` $-X = 1$, while $X = 1$ `iff` $-X = 0$.

Take for example the equation $U = 1$, where U is the formula eq. (2.5) (page 34) of Chapter 2 section 2.2.2 (page 33):

$$(B \wedge -C) \wedge \big((E \vee R) \wedge -(E \wedge R)\big) = 1 \ . \tag{3.20}$$

In Chapter 2 we showed that if you substitute ' 1 ' for ' B ' and ' E ' and ' 0 ' for ' C ' and ' R ', by calculation, the resulting term $U[1/B, 0/C, 1/E, 0/R]$, which is ' $(1 \wedge -0) \wedge ((1 \vee 0) \wedge -(1 \wedge 0))$ ', denotes 1.

Put another way, the following set of equations is a **solution** to the equation $U = 1$:

$$B = 1, \ C = 0, \ E = 1, \ R = 0, \tag{3.21}$$

since,

$$U[1/B, 0/C, 1/E, 0/R] \quad = \quad (1 \wedge -0) \wedge \big((1 \vee 0) \wedge -(1 \wedge 0)\big)$$

[4]Smullyan's practice in (1968)[84], of writing `TX` instead of $X = 1$ and `FX` for $X = 0$ (where X contains no constants), and calling `TX` and `FX` "signed formulas" or "assertions" of the truth or falsity of X, differs from our practice only in notation.

$$\begin{aligned} &= (1 \wedge 1) \wedge (1 \wedge -0) \\ &= 1 \wedge (1 \wedge 1) \\ &= 1 \wedge 1 \\ &= 1 \ . \end{aligned}$$

The solution eq. (3.22) is a set of **compatible** atomic equations, that equate each distinct Boolean variable of L_n ($n \geq 4$) in the set to one and *only* one value. On substituting the constants for these truth values for the Boolean variables in eq. (3.22) to which they were equated, an identity results.

On the other hand, as was also previously shown in section 2.2.2 (page 33), the set

$$B = 1,\ C = 0,\ E = 0,\ R = 0, \tag{3.22}$$

is a solution to the equation $U = 0$, since the term,

$$U[1/B, 0/C, 0/E, 0/R] = (1 \wedge -0) \wedge \big((0 \vee 0) \wedge -(0 \wedge 0)\big) \tag{3.23}$$

denotes 0.

Problems

1. (a) Show that $A = 1,\ B = 1,\ C = 0$ is a solution to the equation: $A \wedge (B \vee C) \Rightarrow A \wedge B = 1$.
   (b) Show that $A = 1,\ B = 1,\ C = 1$ is a solution to the equation: $A \vee (B \wedge C) \Rightarrow A \vee B = 1$.
   (c) Show that $A = 1,\ B = 1$ is a solution to the equation: $A \wedge (B \vee C) \Rightarrow A \wedge B = 1$.
   (d) Show that $A = 0$ is a solution to the equation: $A \wedge (B \vee C) \Rightarrow A \wedge B = 1$.

3.3.2 What Exactly is a Solution to a Set of Boolean Equations?

We are concerned with solutions to sets of simultaneous equations of the form

$$X_1 = c_1, \ldots, X_m = c_m \ , \tag{3.24}$$

where the left side of each equation is a formula of some Boolean language L_n and the right side is a Boolean constant, either ' 0 ' or ' 1 '. A **solution set** Σ for eq. (3.24) is a set of compatible atomic equations of the form

$$\theta_1 = c1, \ldots, \theta_k = c_k \ , \tag{3.25}$$

where $k \leq n$.

Each solution set Σ for eq. (3.24) determines truth value assignments $\sigma : Bv_n \longrightarrow \{0, 1\}$ which assigns each of the k variables θ_i the truth value $d(c_i)$ which c_i denotes. Then Σ is a **solution** of eq. (3.24) `iff` $\tau(X_1) = c_1$ and ... and $\tau(X_m) = c_m$, where $\tau : F(Af) \longrightarrow L_n$ is the Boolean valuation which corresponds to σ. It is a **complete solution** if $\{\theta_1, \ldots, \theta_k\} = Bv_n$.

Problems

1. (a) Which of the solutions given to the equation in the problem at the end of the previous section are complete solutions? WHICH EQUATION ARE WE TALKING ABOUT?

   (b) Is $A = 1,\ B = 1,\ C = 1,\ C = 0$ a solutions? Why or why not? ERROR HERE, C IS DOUBLE VALUED. WHICH EQUATION ARE WE TALKING ABOUT?????

3.3.3 Using Equation Trees to Find Solutions to Simultaneous Boolean Equations, or to Show that None Exist

Now we know what a solution to a set of equations *is*, and how the existence or lack of a solution solves the satisfiability problem and related problems. But how in general are we to go about *finding* the solutions to a set of equations, or showing that there aren't any?

We proceed step by step. First consider the simplest equations in one or two variables, of the form $-A = c$, $A \wedge B = c$, $A \vee B = c$ or $A \Rightarrow B$, where c is a Boolean constant.

The equation $-A = 1$ has a single solution $A = 0$, and the solution to $-A = 0$ is $A = 1$. We may display the results in two **equation trees**:

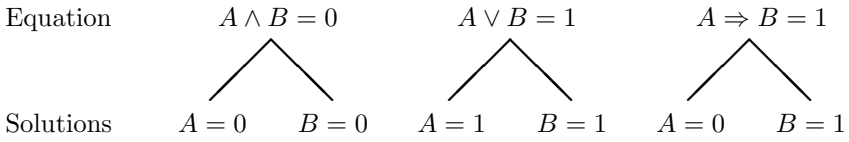
equation	$-A = 1$	$-A = 0$
solution	$A = 0$	$A = 1$

The equation $A \wedge B = 1$ has the solution, $A = 1$, $B = 1$, since $1 \wedge 1 = 1$. It is the only solution, for no solution can contain $A = 0$ or $B = 0$, because $0 \wedge B = A \wedge 0 = 0$. Dually, the equation $A \vee B = 0$ also has a unique solution: $A = 0$, $B = 0$. And since $A \Rightarrow B = 0$ only when $A = 1$ and $B = 0$, $A = 1$, $B = 0$ is the only solution to the equation $A \Rightarrow B = 0$. We also display these results by equation trees:

equation	$A \wedge B = 1$	$A \vee B = 0$	$A \Rightarrow B = 0$
solution	$A = 1$	$A = 0$	$A = 1$
solution	$B = 1$	$B = 0$	$B = 0$

On the other hand, the equation $A \wedge B = 0$ has *two* solutions: $A = 0$ and $B = 0$, since $0 \wedge B = 0$ and $A \wedge 0 = 0$. Neither of these is a **complete** solution in any language L_{n+2}, (which must contain both ' A ' and ' B ') since it does not give the value of each variable ' A ' and ' B ' in ' $A \wedge B$ '. But since each of the equations $A = 0$ and $B = 0$ *is* a solution, it can be extended to a complete solution. For $0 \wedge B = 0$, whether $B = 0$ or $B = 1$, so we have two complete solutions for $A \wedge B = 0$ containing $A = 0$: $A = 0$, $B = 1$ and $A = 0$, $B = 0$. Similarly, $A \wedge 0 = 0$, independently of A, so $B = 0$ belongs to two of the complete solutions to $A \wedge B = 0$, which are: $A = 0$, $B = 0$ and $A = 1$, $B = 0$. This yields three complete solutions in all, each determining a truth value assignment which makes $A \wedge B$ false.

Each of the equations $A \vee B = 1$ and $A \Rightarrow B = 1$ also has two solutions, both of which are incomplete. We display these, along with the two for $A \wedge B = 0$, in these equation trees:

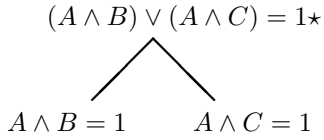


More complicated equations cannot be solved this way in one step. Take for example an equation $X \vee Y = 1$, where X and Y are not both atomic. If X is not atomic, then $X = 1$ is not a solution to the equation $X \vee Y = 1$, because only atomic equations belong to solution sets.

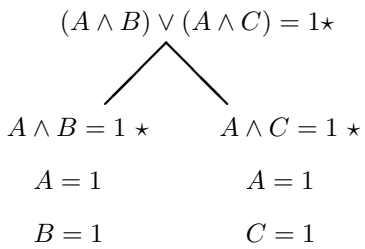
But if it has a solution, then some truth value assignment σ satisfies it, and consequently either σ satisfies X or σ satisfies Y, so that either $X = 1$ has a solution, or $Y = 1$ has a solution. Furthermore any solution to $X = 1$ is a solution to $X \vee Y = 1$ (though generally not complete) and any solution to $Y = 1$ is also a solution to $X \vee Y = 1$.

Thus we can dispense with the equation $X \vee Y = 1$, replacing it with the two equations $X = 1$ and $Y = 1$ in separate branches. As a reminder that $X \vee Y = 1$ has been replaced, you may put a star to the right of it. We also say that the replaced equation has been **fulfilled**.

For instance, suppose that X is $A \wedge B$ and Y is $A \wedge C$. As a first step in constructing an equation tree for $(A \wedge B) \vee (A \wedge C) = 1$, we write:



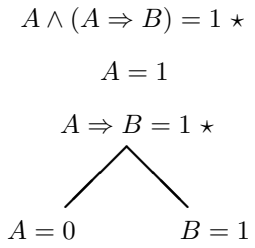
Since $A \wedge B = 1$ and $A \wedge C = 1$ are not atomic equations, neither one is a solution to the **initial** equation at the top of the tree. Our next step is to star these two equations and replace them by their solutions, so that the problem has been solved in two steps:



Each path is **completed** and there is nothing to add to it, since every non-atomic equation has been fulfilled. The atomic equations in each path constitute

a solution to the initial equation. Each solution in this case is incomplete, for the one of the left gives no value for C, so that you can add either $C = 0$ or $C = 1$ to get a complete solution. And on the right, we see that the initial equation holds when $A = 1$ and $C = 1$, independently of the value of B.

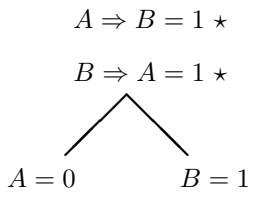
Now let's construct an equation tree from the equation $A \wedge (A \Rightarrow B) = 1$:



Here the set of atomic equations in the left path is not a solution to the initial equation, because it is not compatible, since it has an incompatible subset: the equations $A = 1$ and $A = 0$ **clash** with each other. Any path that contains clashing equations, even if they are not atomic, is **closed**, so the left path is closed. It is also a **completed** path, for nothing more is to be added to it. As a reminder, you can put a ' $\perp$ ' at the bottom of closed path. A path which is not closed is **open**.

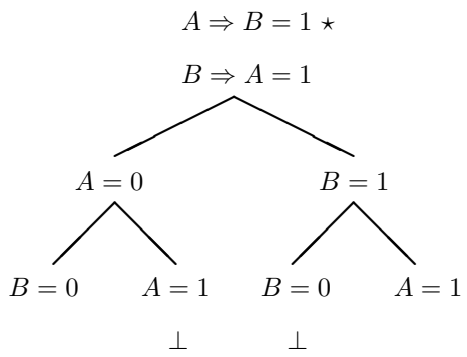
We may have more than one initial equation, which will happen if we want to find a solution to a system of simultaneous equations, or to show that there is no solution. A complete solution to a system of equations provides a value for every variable that appears in the equations.

Here for example a tree constructed from the equations $A \Rightarrow B = 1$, $B \Rightarrow A = 1$. It is a **completed tree**, since every path is completed: it is either closed, or every non-atomic equation in the path has been fulfilled. As our first step we fulfill the top equation:

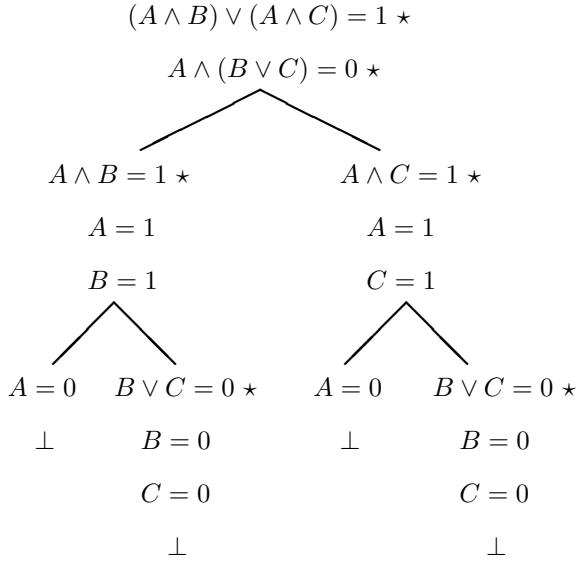


The choice is arbitrary, however. We may fulfill the equations in any order we like.

Next we fulfill the other non-atomic equation. Like the other equation, it has two solutions, which must be put in separate branches at the bottom of both paths, since neither path is closed. Thus we have completed the tree in two steps:



And here is a tree constructed from the set of equations $(A \wedge B) \vee (A \wedge C) = 1$, $A \wedge (B \vee C) = 0$. It is a **closed tree**, since every path is closed, and so also a completed tree. And because it is closed, the system of initial equations has no solution, and therefore no counterexample exists to the argument $(A \wedge B) \vee (A \wedge C) \therefore A \wedge (B \vee C)$. That is, $(A \wedge B) \vee (A \wedge C) \models A \wedge (B \vee C)$:



We summarize the rules for constructing equation trees below. Each rule is **applied** to an equation, the **premise** of the application, which is then fulfilled. The **consequence(s)** of the application are equations, both of which are added to the bottom of each open path, one below the other, if the rule does not produce branches. Otherwise, each of the two equations, added to the bottom of every open path in which their premise occurs, start two new branches, beginning at the junction. We display these rules below. The three dots represent open paths, to the bottom of which the consequences are to be added:

INSERT FIGURE PAGE 154

Observe that the formula on the left hand side of the equation that is a consequence of a rule is an immediate subformula of the formula on the left side of the premise. For instance, X and Y are immediate subformulas of $X \wedge Y$ and of $X \vee Y$.

PROBLEMS

1. Prove by means of an equation tree that:
   $A \vee (B \wedge C) \models (A \vee B)$.

2. Prove by means of an equation tree that:
   $A \wedge (B \vee C) \not\models (A \wedge B)$.

3.3.4 Solving the Basic Problems of Boolean Logic with Equation Trees

You can use equation trees to solve the satisfiability problem, the validity problem and the tautology problem.

1. THE SATISFIABILITY PROBLEM

To determine whether the set $\Gamma = X_1, X_2, \ldots, X_n$ *of formulas is satisfiable:*

Construct a completed equation tree **T** from the initial equations $X_1 = 1$, $X_2 = 1$,...,$X_n = 1$. If **T** is closed, then the set is unsatisfiable. Otherwise, the atomic equations in any open path of **T** constitutes a solution to the initial equations, and determines a truth value assignment and so a valuation satisfying S.

Example 1 The set $A \vee -B$, $B \vee -A$ is satisfiable, since the system of equations $A \vee -B = 1$, $B \vee -A = 1$ has two solutions, $A = B = 1$ and $A = B = 0$, determined by the extreme left and the extreme right paths in this tree:

INSERT FIGURE PAGE 155

Example 2 The set $A \vee -B$, $B \wedge -A$ is unsatisfiable, since the system $A \vee -B = 1$, $B \wedge -A = 1$ has no solution:

INSERT FIGURE PAGE 156

2. THE VALIDITY PROBLEM

To determine whether the argument $X_1, X_2, \ldots, X_n \therefore Y$ *is valid, i.e.* $X_1, X_2, \ldots, X_n \models Y$*:*

Construct a completed equation tree **T** from the initial equations $X_1 = 1$, $X_2 = 1$, ..., $X_n = 1$, $Y = 0$. If **T** is closed, then the argument is valid. Otherwise, any solution to the initial equations, that is, the atomic equations in any open path of **T**, determines a truth value assignment (and corresponding valuation) which is a counterexample to the argument.

Example 1 The atomic equations in the right path of $A \wedge (B \vee C) \not\models A \wedge B$ give a counterexample.

INSERT FIGURE PAGE 156

Example 2 The tree for $A \vee (B \wedge C) \models A \vee B$ is closed:

INSERT FIGURE PAGE 157

3. THE TAUTOLOGY PROBLEM

To determine whether the formula X is a tautology:

Construct a completed tree **T** from $X = 0$. If **T** is closed, $\models X$. If not, the atomic equations in any open path of **T** yield a counterexample.

Example 1 Consider $\not\models A \wedge (B \vee C) \Rightarrow A \wedge B$. The tree that results by adding $A \wedge (B \vee C) \Rightarrow A \wedge B = 0$ to the top of the tree of Example 1 of the Validity Problem is a tree constructed from that equation. The right path again yields a counterexample: $A = 1$, $B = 0$, $C = 1$.

On the other hand, the formula $A \vee (B \wedge C) \Rightarrow A \vee B$ is a tautology, as the equation $A \vee (B \wedge C) \Rightarrow A \vee B = 0$ has no solution, for when added to the top of the tree of Example 2 of the Validity problem, it yields a closed tree constructed from $A \vee (B \wedge C) \Rightarrow A \vee B = 0$.

4. THE EQUIVALENCE PROBLEM

To determine whether X and Y are truth functionally equivalent:

Consider $X \equiv Y$ `iff` both $X \models Y$ and $Y \models X$. The equivalence problem is therefore a special case of the validity problem. Construct two trees: one from $X = 1$, $Y = 0$ and the other from $X = 0$, $Y = 1$. If both trees close, the equation $X = Y$ is an identity. Otherwise, the set of atomic equations in any open path in either tree determines a counterexample to the claim that X and Y are equivalent.

Example 1 Note ' $A \wedge (B \vee C)$ ' is not equivalent to ' $A \wedge B$ ', for as you can see from the tree in Example 1 for the validity problem above, the system of equations,

$$\begin{aligned} A \wedge (B \vee C) &= 1 \,, \\ A \wedge B &= 0 \,, \end{aligned}$$

has the solution $A = 1$, B=0, $C = 1$.

Example 2 Consider that $-A \Rightarrow B \equiv A \vee B$, as we have the close trees:

INSERT FIGURE PAGE 158

3.3.5 Solving the Basic Problems by Truth Value Analysis*

The trouble with proving two formulas equivalent by truth tables, is that you have to calculate the truth values of the formulas for all possible values of the variables they contain. This is a "brute force method".

The method introduce here allows judicious choices that often result in shortcuts, whether the outcome of the test for equivalence is positive or negative. So it is often faster to use than the brute force method, but not always.

Most logicians doubt that there is any such method. Not that this bothers them overly much, for they seldom try to *solve*, say, the satisfiability problem for a particularly large set of formulas. They're more concerned with the nature of the problem than its solution. They may do calculations (preferably by computer) to get at the underlying issues involved. The *real* objection to the brute force method is that it is totally uninformative. It solves the particular problem, if you have enough time, but tells you nothing about the underlying principles.

The procedure we are about to explain is a modification of the method **truth value analysis**, due to Quine[68], which is particularly suitable for solving the equivalence problem by determining whether a given equation is an identity. But as we just saw, any of the other basic problems can be solved if the equivalence problem has been solved. Quine focused on the tautology problem, which also leads to solutions to the other four, and we'll also discuss his method briefly. We start with the equivalence problem.

We first introduce a way to **reduce** a non-atomic open term X which contains a Boolean constant to an equivalent open term which contains no constant, or to a Boolean constant. The open term Y to which X reduces is itself **irreducible**. We reduce X to Y in one or more steps by performing a series of **reductions** taken from the following table:

Put	for	or for	or for	Because:
X	$X \wedge 1$	$X \vee 0$	$1 \Rightarrow X$	$X \equiv X \wedge 1 \equiv X \vee 0 \equiv 1 \Rightarrow X$
X	$1 \wedge X$	$0 \vee X$	$--X$	$X \equiv 1 \wedge X \equiv 0 \vee X \equiv --X$
$-X$	$X \Rightarrow 0$			$X \Rightarrow 0 \equiv -X$
1	$X \vee 1$	$1 \vee X$	$X \Rightarrow 1$	$1 \equiv X \vee 1 \equiv 1 \vee X \equiv X \Rightarrow 1$
1	-0	$0 \Rightarrow X$	$X \Rightarrow X$	$1 \equiv -0 \equiv 0 \Rightarrow X \equiv X \Rightarrow X$
1		$X \vee -X$	$-X \vee X$	$1 \equiv -X \vee X \equiv X \vee -X$
0	-1	$0 \wedge X$	$X \wedge 0$	$0 \equiv -1 \equiv 0 \wedge X \equiv X \wedge 0$
0		$X \wedge -X$	$-X \wedge X$	$0 \equiv X \wedge -X \equiv -X \wedge X$

*This part may be skipped without loss of continuity.

For instance, ‘ $(1 \wedge B) \vee (1 \wedge C)$ ’ reduces to ‘ $B \vee C$ ’, for

$$(1 \wedge B) \vee (1 \wedge C) = B \vee C$$

while ‘ $(1 \wedge B) \vee (1 \wedge C)$ ’ reduces to ‘ $1 \vee 1$ ’ which in turn reduces to 1, since,

$$\begin{aligned} (1 \wedge B) \vee (1 \wedge C) &= 1 \wedge 1 \\ &= 1 \end{aligned}$$

Of course, as we saw in Chapter 2, every term reduces to a constant, and the reduction procedure in that case amounts to an ordinary calculation.

To test a pair X, Y of formulas for equivalence by the method of truth value analysis, we start an **analysis tree** with the equation $X = Y$. We choose a variable θ which occurs in X, Y and below it put $X[1/\theta] = Y[1/\theta]$ on the left and $X[0/\theta] = Y[0/\theta]$ on the right. Then using the above rules, in one or more steps put new equations below the equations on each side, until you get a **reduced equation** $X_1 = Y_1$ on the left and $X_0 = Y_0$ on the right. Here $X[1/\theta]$ reduces to X_1, $X[0/\theta]$ reduces to X_0, $Y[1/\theta]$ reduces to Y_1, and $Y[0/\theta]$ reduces to Y_0.[5] This gives us the tree

INSERT FIGURE page 160

We repeat the process as often as necessary. Every **path** through the tree, starting at the top of the tree and continuing to the bottom, eventually ends in an identity $X = X$, or in an “anti-identity” of the form $-X = X$, $X = -X$, $1 = 0$ or $0 = 1$. At this point the tree is completed, and if there is an identity at the bottom of every path, then $X \equiv Y$. Otherwise X and Y are not equivalent.

Here, for example is an analysis tree for the equation,

$$-(A \wedge B) = -A \vee -B$$

which verifies that it is an identity, since every path through the tree ends in an identity:

INSERT FIGURE page 160

If an equation is not an identity, you can find a **counter-example** to the claim that it’s an identity, i.e. a truth value assignment in which the sides are not equal. To do this, determine what substitutions of truth values for which variables led to an anti-identity. For example, an analysis tree for the equation $-(A \wedge B) = -A \wedge -B$ is

INSERT FIGURE page 160

The middle two paths end in anti-identities and we put a ‘ $\top$ ’ below each one of them to single them out. In the path to the right of the leftmost path, the substitutions which reduced to ‘ $0 = -0$ ’ were determined by the equations

[5] You can choose the variables for which to substitute ‘ 1 ’ and ‘ 0 ’ in any order. The method works, because the inference from one identity to another is reversible.

$A = 1$, $B = 0$, while in the path to the left of the rightmost path, the substitutions were given by $A = 0$, $B = 1$. Thus there are two counter-examples:

$$\begin{array}{ll} \text{`}A\text{'} \longrightarrow 1 & \text{`}A\text{'} \longrightarrow 0 \\ \text{`}B\text{'} \longrightarrow 0 & \text{`}B\text{'} \longrightarrow 1 \end{array}$$

(Check them out!)

Quine himself concentrated on the validity and satisfiability problems, which he recast as tautology problems, and then reduced open terms rather than equations. A formula which under successive substitutions of both 1 and 0 for one or more variables is a tautology. For example, to prove that $-A \vee B \models A \Rightarrow B$, reduce $-A \vee B \Rightarrow (A \Rightarrow B)$ to 1:

INSERT FIGURE page 161

In this case, a further short-cut is available (Quine's "full sweep"): Since only one truth value assignment σ to ' A ' and ' B ' falsifies ' $A \Rightarrow B$ ', i.e. $\sigma(\text{`}A\text{'}) = 1$ and $\sigma(\text{`}B\text{'}) = 0$, substitute ' 1 ' for ' A ' and ' 0 ' for ' B ' in ' $-A \vee B$ '. If the resulting term reduces to ' 0 ', then $-A \vee B \models A \Rightarrow B$:

$$\begin{array}{rcl} -1 & \vee & 0 \\ 0 & \vee & 0 \\ & 0 & \end{array}$$

PROBLEMS

1. (a) Show by truth value analysis that,
   $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$.
   (b) Prove the dual:
   $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.
2. Show in one full sweep that $-A \Rightarrow B \models A \vee B$.