1 Classical Logic and the Material Conditional

1.1 Introduction

1.1.1 The first purpose of this chapter is to review classical propositional logic, including semantic tableaux. The chapter also sets out some basic terminology and notational conventions for the rest of the book.

1.1.2 In the second half of the chapter we also look at the notion of the conditional that classical propositional logic gives, and, specifically, at some of its shortcomings.

1.1.3 The point of logic is to give an account of the notion of validity: what follows from what. Standardly, validity is defined for inferences couched in a formal language, a language with a well-defined vocabulary and grammar, the object language. The relationship of the symbols of the formal language to the words of the vernacular, English in this case, is always an important issue.

1.1.4 Accounts of validity themselves are in a language that is normally distinct from the object language. This is called the metalanguage. In our case, this is simply mathematical English. Note that ‘iff’ means ‘if and only if’.

1.1.5 It is also standard to define two notions of validity. The first is semantic. A valid inference is one that preserves truth, in a certain sense. Specifically, every interpretation (that is, crudely, a way of assigning truth values) that makes all the premises true makes the conclusion true. We use the metalinguistic symbol ‘|=’ for this. What distinguishes different logics is the different notions of interpretation they employ.
1.1.6 The second notion of validity is proof-theoretic. Validity is defined in terms of some purely formal procedure (that is, one that makes reference only to the symbols of the inference). We use the metalinguistic symbol ‘|=’ for this notion of validity. In our case, this procedure will (mainly) be one employing tableaux. What distinguish different logics here are the different tableau procedures employed.

1.1.7 Most contemporary logicians would take the semantic notion of validity to be more fundamental than the proof-theoretic one, though the matter is certainly debatable. However, given a semantic notion of validity, it is always useful to have a proof-theoretic notion that corresponds to it, in the sense that the two definitions always give the same answers. If every proof-theoretically valid inference is semantically valid (so that |= entails |=) the proof-theory is said to be sound. If every semantically valid inference is proof-theoretically valid (so that |= entails |=) the proof-theory is said to be complete.

### 1.2 The Syntax of the Object Language

1.2.1 The symbols of the object language of the propositional calculus are an infinite number of propositional parameters: \(p_0, p_1, p_2, \ldots\); the connectives: ¬ (negation), ∧ (conjunction), ∨ (disjunction), ⊃ (material conditional), ≡ (material equivalence); and the punctuation marks: (, ).

1.2.2 The (well-formed) formulas of the language comprise all, and only, the strings of symbols that can be generated recursively from the propositional parameters by the following rule:

If \(A\) and \(B\) are formulas, so are \(\neg A\), \((A \lor B)\), \((A \land B)\), \((A \supset B)\), \((A \equiv B)\).

1.2.3 I will explain a number of important notational conventions here. I use capital Roman letters, \(A, B, C, \ldots\), to represent arbitrary formulas of the object language. Lower-case Roman letters, \(p, q, r, \ldots\), represent arbitrary, \(^1\) These are often called ‘propositional variables’.
but distinct, propositional parameters. I will always omit outermost parentheses of formulas if there are any. So, for example, I write \((A \supset (B \lor \neg C))\) simply as \(A \supset (B \lor \neg C)\). Upper-case Greek letters, \(\Sigma, \Pi, \ldots\), represent arbitrary sets of formulas; the empty set, however, is denoted by the (lower case) \(\phi\), in the standard way. I often write a finite set, \(\{A_1, A_2, \ldots, A_n\}\), simply as \(A_1, A_2, \ldots, A_n\).

### 1.3 Semantic Validity

1.3.1 An interpretation of the language is a function, \(\nu\), which assigns to each propositional parameter either 1 (true), or 0 (false). Thus, we write things such as \(\nu(p) = 1\) and \(\nu(q) = 0\).

1.3.2 Given an interpretation of the language, \(\nu\), this is extended to a function that assigns every formula a truth value, by the following recursive clauses, which mirror the syntactic recursive clauses:

\[
\begin{align*}
\nu(\neg A) &= 1 \text{ if } \nu(A) = 0, \text{ and } 0 \text{ otherwise.} \\
\nu(A \land B) &= 1 \text{ if } \nu(A) = \nu(B) = 1, \text{ and } 0 \text{ otherwise.} \\
\nu(A \lor B) &= 1 \text{ if } \nu(A) = 1 \text{ or } \nu(B) = 1, \text{ and } 0 \text{ otherwise.} \\
\nu(A \supset B) &= 1 \text{ if } \nu(A) = 0 \text{ or } \nu(B) = 1, \text{ and } 0 \text{ otherwise.} \\
\nu(A \equiv B) &= 1 \text{ if } \nu(A) = \nu(B), \text{ and } 0 \text{ otherwise.}
\end{align*}
\]

1.3.3 Let \(\Sigma\) be any set of formulas (the premises); then \(A\) (the conclusion) is a semantic consequence of \(\Sigma\) (\(\Sigma \models A\)) iff there is no interpretation that makes all the members of \(\Sigma\) true and \(A\) false, that is, every interpretation that makes all the members of \(\Sigma\) true makes \(A\) true. ‘\(\Sigma \not\models A\)’ means that it is not the case that \(\Sigma \models A\).

1.3.4 \(A\) is a logical truth (tautology) (\(\models A\)) iff it is a semantic consequence of the empty set of premises (\(\phi \models A\)), that is, every interpretation makes \(A\) true.

\[\begin{array}{c|cc}
\land & 1 & 0 \\
\hline
1 & 1 & 0 \\
0 & 0 & 0
\end{array}\]

2 The reader might be more familiar with the information contained in these clauses when it is depicted in the form of a table, usually called a truth table, such as the one for conjunction displayed:
1.4 Tableaux

1.4.1 A tree is a structure that looks, generally, like this:\textsuperscript{3}

\[
\begin{array}{c}
\vdots \\
\downarrow \\
\vdots \\
\downarrow \\
\vdots \\
\end{array}
\]

The dots are called nodes. The node at the top is called the root. The nodes at the bottom are called tips. Any path from the root down a series of arrows as far as you can go is called a branch. (Later on we will have trees with infinite branches, but not yet.)

1.4.2 To test an inference for validity, we construct a tableau which begins with a single branch at whose nodes occur the premises (if there are any) and the negation of the conclusion. We will call this the initial list. We then apply rules which allow us to extend this branch. The rules for the conditional are as follows:

\[
\begin{array}{c}
A \supset B \\
\downarrow \\
\neg A \\
\downarrow \\
\neg B \\
\end{array}
\]

\[
\begin{array}{c}
\neg(A \supset B) \\
\downarrow \\
\neg A \\
\downarrow \\
\neg B \\
\end{array}
\]

The rule on the right is to be interpreted as follows. If we have a formula \((\neg(A \supset B))\) at a node, then every branch that goes through that node is extended with two further nodes, one for \(A\) and one for \(\neg B\). The rule on the left is interpreted similarly: if we have a formula \(A \supset B\) at a node, then every branch that goes through that node is split at its tip into two branches; one contains a node for \(\neg A\); the other contains a node for \(B\).

\textsuperscript{3} Strictly speaking, for those who want the precise mathematical definition, it is a partial order with a unique maximum element, \(x_0\), such that for any element, \(x_n\), there is a unique finite chain of elements \(x_n \leq x_{n-1} \leq \cdots \leq x_1 \leq x_0\).
1.4.3 For example, to test the inference whose premises are \( A \supset B, B \supset C, \) and whose conclusion is \( A \supset C, \) we construct the following tree:

\[
\begin{align*}
A & \supset B \\
\downarrow \\
B & \supset C \\
\downarrow \\
\neg(A \supset C) \\
\downarrow \\
A \\
\downarrow \\
\neg C
\end{align*}
\]

The first three formulas are the premises and negated conclusion. The next two formulas are produced by the rule for the negated conditional applied to the negated conclusion; the first split on the branch is produced by applying the rule for the conditional to the first premise; the next splits are produced by applying the same rule to the second premise. (Ignore the ‘\( \times \)’s: we will come back to those in a moment.)

1.4.4 The other connectives also have rules, which are as follows.

\[
\begin{align*}
\neg\neg A \\
\downarrow \\
A
\end{align*}
\]
An Introduction to Non-Classical Logic

Intuitively, what a tableau means is the following. If we apply a rule to a formula, then if that formula is true in an interpretation, so are the formulas below on at least one of the branches that the rule generates. (Of course, there may be only one such branch.) This is a useful mnemonic for remembering the rules. It must be stressed, though, that officially the rules are purely formal.

1.4.5 A tableau is complete iff every rule that can be applied has been applied. By applying the rules over and over, we may always construct a complete tableau. In the present case, the branches of a completed tableau are always finite, but in the tableaux of some subsequent chapters they may be infinite.

1.4.6 A branch is closed iff there are formulas of the form $A$ and $\neg A$ on two of its nodes; otherwise it is open. A closed branch is indicated by writing an $\times$ at the bottom. A tableau itself is closed iff every branch is closed; otherwise it is open. Thus the tableau of 1.4.3 is closed: the leftmost branch contains $A$ and $\neg A$; the next contains $A$ and $\neg A$ (and $C$ and $\neg C$); the next contains $B$ and $\neg B$; the rightmost contains $C$ and $\neg C$.

1.4.7 $\phi$ is a proof-theoretic consequence of the set of formulas $\Sigma (\Sigma \vdash \phi)$ iff there is a complete tree whose initial list comprises the members of $\Sigma$ and the negation of $A$, and which is closed. We write $\vdash \phi$ to mean that $\phi \vdash A$.

---

$\neg (A \land B)$ $A \land B$

$\neg A$ $\neg B$ $A$

$\neg (A \equiv B)$ $A \equiv B$

$\neg A$ $A$

$\neg \neg B$ $B$
that is, where the initial list of the tableau comprises just $\neg A$. ‘$\Sigma \not\vdash A$’ means that it is not the case that $\Sigma \vdash A$.\footnote{There may, in fact, be several completed trees for an inference, depending upon the order of the premises in the initial list and the order in which rules are applied. Fortunately, they all give the same result, though this is not entirely obvious. See 1.14, problem 5.}

1.4.8 Thus, the tree of 1.4.3 shows that $A \supset B$, $B \supset C \vdash A \supset C$. Here is another, to show that $\vdash ((A \supset B) \land (A \supset C)) \supset (A \supset (B \land C))$. To save space, we omit arrows where a branch does not divide.

\[
\neg(((A \supset B) \land (A \supset C)) \supset (A \supset (B \land C)))
\]
\[
(A \supset B) \land (A \supset C)
\]
\[
\neg(A \supset (B \land C))
\]
\[
(A \supset B)
\]
\[
(A \supset C)
\]
\[
A
\]
\[
\neg(B \land C)
\]
\[
\neg B
\]
\[
\neg C
\]
\[
\neg A
\]
\[
B
\]
\[
\neg A
\]
\[
B
\]
\[
\times
\]
\[
\times
\]
\[
\times
\]
\[
\times
\]
\[
\neg A
\]
\[
C
\]
\[
\times
\]
\[
\times
\]

Note that when we find a contradiction on a branch, there is no point in continuing it further. We know that the branch is going to close, whatever else is added to it. Hence, we need not bother to extend a branch as soon as it is found to close. Notice also that, wherever possible, we apply rules that do not split branches before rules that split branches. Though this is not essential, it keeps the tableau simpler, and is therefore useful practically.

1.4.9 In practice, it is also a useful idea to put a tick at the side of a formula once one has applied a rule to it. Then one knows that one can forget about it.
7 Many-valued Logics

7.1 Introduction

7.1.1 In this chapter, we leave possible-world semantics for a time, and turn to the subject of propositional many-valued logics. These are logics in which there are more than two truth values.

7.1.2 We have a look at the general structure of a many-valued logic, and some simple but important examples of many-valued logics. The treatment will be purely semantic: we do not look at tableaux for the logics, nor at any other form of proof procedure. Tableaux for some many-valued logics will emerge in the next chapter.

7.1.3 We also look at some of the philosophical issues that have motivated many-valued logics, how many-valuedness affects the issue of the conditional, and a few other noteworthy issues.

7.2 Many-valued Logic: The General Structure

7.2.1 Let us start with the general structure of a many-valued logic. To simplify things, we take, henceforth, \( A \equiv B \) to be defined as \((A \supset B) \land (B \supset A)\).

7.2.2 Let \( \mathcal{C} \) be the class of connectives of classical propositional logic \( \{\land, \lor, \neg, \supset\} \). The classical propositional calculus can be thought of as defined by the structure \( \langle \mathcal{V}, \mathcal{D}, \{f_c; c \in \mathcal{C}\} \rangle \). \( \mathcal{V} \) is the set of truth values \( \{1,0\} \). \( \mathcal{D} \) is the set of designated values \( \{1\} \); these are the values that are preserved in valid inferences. For every connective, \( c, f_c \) is the truth function it denotes. Thus, \( f_\neg \) is a one-place function such that \( f_\neg(0) = 1 \) and \( f_\neg(1) = 0 \); \( f_\land \) is a two-place function such that \( f_\land(x,y) = 1 \) if \( x = y = 1 \), and \( f_\land(x,y) = 0 \) otherwise; and so
on. These functions can be (and often are) depicted in the following ‘truth tables’.

<table>
<thead>
<tr>
<th>( f )</th>
<th>( \neg )</th>
<th>( f )</th>
<th>( \land )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

7.2.3 An interpretation, \( \nu \), is a map from the propositional parameters to \( \mathcal{V} \). An interpretation is extended to a map from all formulas into \( \mathcal{V} \) by applying the appropriate truth functions recursively. Thus, for example, \( \nu(\neg(p \land q)) = f_\neg(\nu(p \land q)) = f_\neg(f_\land(\nu(p), \nu(q))) \). (So if \( \nu(p) = 1 \) and \( \nu(q) = 0 \), \( \nu(\neg(p \land q)) = f_\neg(f_\land(1, 0)) = f_\neg(0) = 1 \).) Finally, an inference is semantically valid just if there is no interpretation that assigns all the premises a value in \( \mathcal{D} \), but assigns the conclusion a value not in \( \mathcal{D} \).

7.2.4 A many-valued logic is a natural generalisation of this structure. Given some propositional language with connectives \( \mathcal{C} \) (maybe the same as those of the classical propositional calculus, maybe different), a logic is defined by a structure \( (\mathcal{V}, \mathcal{D}, \{f_c; c \in \mathcal{C}\}) \). \( \mathcal{V} \) is the set of truth values: it may have any number of members (\( \geq 1 \)). \( \mathcal{D} \) is a subset of \( \mathcal{V} \), and is the set of designated values. For every connective, \( c \), \( f_c \) is the corresponding truth function. Thus, if \( c \) is an \( n \)-place connective, \( \nu(c(A_1, \ldots, A_n)) = f_c(\nu(A_1), \ldots, \nu(A_n)) \). Finally, \( \Sigma \models A \) iff there is no interpretation, \( \nu \), such that for all \( B \in \Sigma \), \( \nu(B) \in \mathcal{D} \), but \( \nu(A) \notin \mathcal{D} \). \( A \) is a logical truth iff \( \phi \models A \), i.e., iff for every interpretation \( \nu(A) \in \mathcal{D} \).

7.2.5 An interpretation for the language is a map, \( \nu \), from propositional parameters into \( \mathcal{V} \). This is extended to a map from all formulas of the language to \( \mathcal{V} \) by applying the appropriate truth functions recursively. Thus, if \( c \) is an \( n \)-place connective, \( \nu(c(A_1, \ldots, A_n)) = f_c(\nu(A_1), \ldots, \nu(A_n)) \). Finally, \( \Sigma \models A \) iff there is no interpretation, \( \nu \), such that for all \( B \in \Sigma \), \( \nu(B) \in \mathcal{D} \), but \( \nu(A) \notin \mathcal{D} \). \( A \) is a logical truth iff \( \phi \models A \), i.e., iff for every interpretation \( \nu(A) \in \mathcal{D} \).

7.2.6 If \( \mathcal{V} \) is finite, the logic is said to be finitely many-valued. If \( \mathcal{V} \) has \( n \) members, it is said to be an \( n \)-valued logic.

7.2.7 For any finitely many-valued logic, the validity of an inference with finitely many premises can be determined, as in the classical propositional calculus, simply by considering all the possible cases. We list all the possible combinations of truth values for the propositional parameters employed.
Then, for each combination, we compute the value of each premise and the conclusion. If, in any of these, the premises are all designated and the conclusion is not, the inference is invalid. Otherwise, it is valid. We will have an example of this procedure in the next section.

7.2.8 This method, though theoretically adequate, is often impractical because of exponential explosion. For if there are \( m \) propositional parameters employed in an inference, and \( n \) truth values, there are \( n^m \) possible cases to consider. This grows very rapidly. Thus, if the logic is 4-valued and we have an inference involving just four propositional parameters, there are already 256 cases to consider!

### 7.3 The 3-valued Logics of Kleene and Łukasiewicz

7.3.1 In what follows, we consider some simple examples of the above general structure. All the examples that we consider are 3-valued logics. The language, in every case, is that of the classical propositional calculus.

7.3.2 A simple example of a 3-valued logic is as follows. \( \mathcal{V} = \{1, i, 0\} \). 1 and 0 are to be thought of as true and false, as usual. \( i \) is to be thought of as neither true nor false. \( \mathcal{D} \) is just \( \{1\} \). The truth functions for the connectives are depicted as follows:

\[
\begin{array}{c|c|c|c}
\hline
f_\neg & f_\wedge & f_\lor & f_\supset \\
\hline
1 & 0 & 1 & 1 \\
i & i & i & i \\
0 & 1 & 0 & 0 \\
\hline
\end{array}
\]

Thus, if \( \nu(p) = 1 \) and \( \nu(q) = i \), \( \nu(\neg p) = 0 \) (top row of \( f_\neg \)), \( \nu(\neg p \lor q) = i \) (bottom row, middle column of \( f_\lor \)), etc.

7.3.3 Note that if the inputs of any of these functions are classical (1 or 0), the output is exactly the same as in the classical case. We compute the other entries as follows. Take \( A \wedge B \) as an example. If \( A \) is false, then, whatever \( B \) is, this is (classically) sufficient to make \( A \wedge B \) false. In particular, if \( B \) is neither true nor false, \( A \wedge B \) is false. If \( A \) is true, on the other hand, and \( B \) is neither true nor false, there is insufficient information to compute the (classical) value of \( A \wedge B \); hence, \( A \wedge B \) is neither true nor false. Similar reasoning justifies all the other entries.
7.3.4 The logic specified above is usually called the (strong) Kleene 3-valued logic, often written $K_3$.\(^1\)

7.3.5 The following table verifies that $p \supset q \models_{K_3} \neg q \supset \neg p$:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \supset q$</th>
<th>$\neg q \supset \neg p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>$i$</td>
<td>$i$</td>
<td>$i$ 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>$i$</td>
<td>1</td>
<td>1</td>
<td>0 1 $i$</td>
</tr>
<tr>
<td>$i$</td>
<td>$i$</td>
<td>$i$</td>
<td>$i$ $i$ $i$</td>
</tr>
<tr>
<td>$i$</td>
<td>0</td>
<td>$i$</td>
<td>1 $i$ $i$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>$i$</td>
<td>1</td>
<td>$i$ 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

In the last three columns, the first number is the value of $\neg q$; the last number is that of $\neg p$, and the central number (printed in bold) is the value of the whole formula. As can be seen, there is no interpretation where the premise is designated, that is, has the value 1, and the conclusion is not.

7.3.6 In checking for validity, it may well be easier to work backwards. Consider the formula $p \supset (q \supset p)$. Suppose that this is undesignated. Then it has either the value 0 or the value $i$. If it has the value 0, then $p$ has the value 1 and $q \supset p$ has the value 0. But if $p$ has the value 1, so does $q \supset p$. This situation is therefore impossible. If it has the value $i$, there are three possibilities:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q \supset p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$i$</td>
</tr>
<tr>
<td>$i$</td>
<td>$i$</td>
</tr>
<tr>
<td>$i$</td>
<td>0</td>
</tr>
</tbody>
</table>

The first case is not possible, since if $p$ has the value 1, so does $q \supset p$. Nor is the last case, since if $p$ has the value $i$, $q \supset p$ has value either $i$ or 1. But the

---

\(^1\) Weak Kleene logic is the same as $K_3$, except that, for every truth function, if any input is $i$, so is the output.
second case is possible, namely when both \( p \) and \( q \) have the value \( i \). Thus, \( \nu(p) = \nu(q) = i \) is a counter-model to \( p \supset (q \supset p) \), as a truth-table check confirms. So \( \not\models \ K_3 \ p \supset (q \supset p) \).

7.3.7 A distinctive thing about \( K_3 \) is that the law of excluded middle is not valid: \( \not\models \ K_3 \ p \lor \neg p \). (Counter-model: \( \nu(p) = i \).) However, \( K_3 \) is distinct from intuitionist logic. As we shall see in 7.10.8, intuitionist logic is not the same as any finitely many-valued logic.

7.3.8 In fact, \( K_3 \) has no logical truths at all (7.14, problem 3)! In particular, the law of identity is not valid: \( \not\models \ K_3 \ p \supset p \). (Simply give \( p \) the value \( i \).) This may be changed by modifying the middle entry of the truth function for \( \supset \), so that \( f_{\supset} \) becomes:

\[
\begin{array}{cccc}
f_{\supset} & 1 & i & 0 \\
1 & 1 & i & 0 \\
i & 1 & 1 & i \\
0 & 1 & 1 & 1 \\
\end{array}
\]

(The meaning of \( A \supset B \) in \( K_3 \) can still be expressed by \( \neg A \lor B \), since this has the same truth table, as may be checked.) Now, \( A \supset A \) always takes the value 1.

7.3.9 The logic resulting from this change is one originally given by Łukasiewicz, and is often called \( L_3 \).

7.4 \( LP \) and \( RM_3 \)

7.4.1 Another 3-valued logic is the one often called \( LP \). This is exactly the same as \( K_3 \), except that \( D = \{1, i\} \).

7.4.2 In the context of \( LP \), the value \( i \) is thought of as both true and false. Consequently, 1 and 0 have to be thought of as true and true only, and false and false only, respectively. This change does not affect the truth tables, which still make perfectly good sense under the new interpretation. For example, if \( A \) takes the value 1 and \( B \) takes the value \( i \), then \( A \) and \( B \) are both true; hence, \( A \land B \) is true; but since \( B \) is false, \( A \land B \) is false. Hence, the value of \( A \land B \) is \( i \). Similarly, if \( A \) takes the value 0, and \( B \) takes the value \( i \), then \( A \)}
and $B$ are both false, so $A \land B$ is false; but only $B$ is true, so $A \land B$ is not true. Hence, $A \land B$ takes the value 0.

7.4.3 However, the change of designated values makes a crucial difference. For example, $\models_{LP} p \lor \neg p$. (Whatever value $p$ has, $p \lor \neg p$ takes either the value 1 or $i$. Thus it is always designated.) This fails in $K_3$, as we saw in 7.3.7.

7.4.4 On the other hand, $p \land \neg p \not\models_{LP} q$. Counter-model: $\nu(p) = i$ (making $\nu(p \land \neg p) = i$), $\nu(q) = 0$. But $p \land \neg p$ can never take the value 1 and so be designated in $K_3$. Thus, the inference is valid in $K_3$.

7.4.5 A notable feature of $LP$ is that modus ponens is invalid: $p, p \supset q \not\models_{LP} q$. (Assign $p$ the value $i$, and $q$ the value 0.)

7.4.6 One way to rectify this is to change the truth function for $\supset$ to the following:

<table>
<thead>
<tr>
<th>$f_{\supset}$</th>
<th>1</th>
<th>$i$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$i$</td>
<td>1</td>
<td>$i$</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(As in 7.3.8, the meaning of $A \supset B$ in $LP$ can still be expressed by $\neg A \lor B$.) Now, if $A$ and $A \supset B$ have designated values (1 or $i$), so does $B$, as a moment checking the truth table verifies.

7.4.7 This change gives the logic often called $RM_3$.

### 7.5 Many-valued Logics and Conditionals

7.5.1 Further details of the properties of $\land$, $\lor$ and $\neg$ in the logics we have just met will emerge in the next chapter. For the present, let us concentrate on the conditional.

7.5.2 In past chapters, we have met a number of problematic inferences concerning conditionals. The following table summarises whether or not they hold in the various logics we have looked at. (A tick means yes; a cross means no.)
8 First Degree Entailment

8.1 Introduction

8.1.1 In this chapter we look at a logic called first degree entailment (FDE). This is formulated, first, as a logic where interpretations are relations between formulas and standard truth values, rather than as the more usual functions. Connections between FDE and the many-valued logics of the last chapter will emerge.

8.1.2 We also look at an alternative possible-world semantics for FDE, which will introduce us to a new kind of semantics for negation.

8.1.3 Finally, we look at the relation of all this to the explosion of contradictions, and to the disjunctive syllogism.

8.2 The Semantics of FDE

8.2.1 The language of FDE contains just the connectives $\land$, $\lor$ and $\neg$. $A \supset B$ is defined, as usual, as $\neg A \lor B$.

8.2.2 In the classical propositional calculus, an interpretation is a function from formulas to the truth values 0 and 1, written thus: $\nu(A) = 1$ (or 0). Packed into this formalism is the assumption (usually made without comment in elementary logic texts) that every formula is either true or false; never neither, and never both.

8.2.3 As we saw in the last chapter, there are reasons to doubt this assumption. If one does, it is natural to formulate an interpretation, not as a function, but as a relation between formulas and truth values. Thus, a formula may relate to 1; it may relate to 0; it may relate to both; or it may relate to neither. This is the main idea behind the following semantics for FDE.
8.2.4 Note that it is now very important to distinguish between being false in an interpretation and not being true in it. (There is, of course, no difference in the classical case.) The fact that a formula is false (relates to 0) does not mean that it is untrue (it may also relate to 1). And the fact that it is untrue (does not relate to 1) does not mean that it is false (it may not relate to 0 either).

8.2.5 An FDE interpretation is a relation, $\rho^1$ between propositional parameters and the values 1 and 0. (In mathematical notation, $\rho \subseteq P \times \{1, 0\}$, where $P$ is the set of propositional parameters.) Thus, $p \rho 1$ means that $p$ relates to 1, and $p \rho 0$ means that $p$ relates to 0.

8.2.6 Given an interpretation, $\rho$, this is extended to a relation between all formulas and truth values by the recursive clauses:

- $A \land B \rho 1$ iff $A \rho 1$ and $B \rho 1$
- $A \land B \rho 0$ iff $A \rho 0$ or $B \rho 0$
- $A \lor B \rho 1$ iff $A \rho 1$ or $B \rho 1$
- $A \lor B \rho 0$ iff $A \rho 0$ and $B \rho 0$
- $\neg A \rho 1$ iff $A \rho 0$
- $\neg A \rho 0$ iff $A \rho 1$

Note that these are exactly the same as the classical truth conditions, stripped of the assumption that truth and falsity are exclusive and exhaustive. Thus, a conjunction is true (under an interpretation) if both conjuncts are true (under that interpretation); it is false if at least one conjunct is false, etc.

8.2.7 As an example of how these conditions work, consider the formula $\neg p \land (q \lor r)$. Suppose that $p \rho 1$, $p \rho 0$, $q \rho 1$ and $r \rho 0$, and that $\rho$ relates no parameter to anything else. Since $p$ is true, $\neg p$ is false; and since $p$ is false, $\neg p$ is true. Thus $\neg p$ is both true and false. Since $q$ is true, $q \lor r$ is true; and since $q$ is not false, $q \lor r$ is not false. Thus, $q \lor r$ is simply true. But then, $\neg p \land (q \lor r)$ is true, since both conjuncts are true; and false, since the first conjunct is false. That is, $\neg p \land (q \lor r) \rho 1$ and $\neg p \land (q \lor r) \rho 0$.

\[1\] Not to be confused with the reflexive $\rho$ of normal modal logics.
8.2.8 Semantic consequence is defined, in the usual way, in terms of truth preservation, thus:

\[ \Sigma \models A \text{ iff for every interpretation, } \rho, \text{ if } B \rho \text{ for all } B \in \Sigma \text{ then } A \rho \text{1} \]

and:

\[ \models A \text{ iff } \phi \models A, \text{ i.e., for all } \rho, A \rho \text{1} \]

8.3 Tableaux for FDE

8.3.1 Tableaux for FDE can be obtained by modifying those for the classical propositional calculus as follows.

8.3.2 Each entry of the tableau is now of the form \( A, + \) or \( A, - \). Intuitively, \( A, + \) means that \( A \) is true, \( A, - \) means that it isn’t. As we noted in 8.2.4, and as with intuitionist logic (6.4.1), \( \neg A, + \) no longer means the same, intuitively, as \( A, - \).

8.3.3 To test the claim that \( A_1, \ldots, A_n \vdash B \), we start with an initial list of the form:

\[
\begin{align*}
A_1, + \\
\vdots \\
A_n, + \\
B, - 
\end{align*}
\]

8.3.4 The tableaux rules are as follows:

\[
\begin{array}{c}
A \land B, + \\
\downarrow \\
A, + \\
B, + \\
A \lor B, + \\
\swarrow \searrow \\
A, + \\
B, + \\
\downarrow \\
A, - \\
B, - \\
\neg (A \land B), + \\
\downarrow \\
\neg A \lor \neg B, + \\
\end{array}
\]

\[
\begin{array}{c}
A \land B, - \\
\swarrow \searrow \\
A, - \\
B, - \\
A \lor B, - \\
\downarrow \\
\neg A \lor \neg B, - \\
\end{array}
\]
The first two rules speak for themselves: if \( A \land B \) is true, \( A \) and \( B \) are true; if \( A \land B \) is not true, then one or other of \( A \) and \( B \) is not true. Similarly for the rules for disjunction. The other rules are also easy to remember, since \( \neg(A \land B) \) and \( \neg A \lor \neg B \) have the same truth values in FDE, as do \( \neg(A \lor B) \) and \( \neg A \land \neg B \), and \( \neg \neg A \) and \( A \). (De Morgan’s laws and the law of double negation, respectively.)

8.3.5 Finally, a branch of a tableau closes if it contains nodes of the form \( A, + \) and \( A, - \).

8.3.6 For example, the following tableau demonstrates that \( \neg(B \land \neg C) \land A \vdash (\neg B \lor C) \lor D \):

\[
\begin{align*}
\neg(B \land \neg C) \land A, + \\
(\neg B \lor C) \lor D, - \\
\neg(B \land \neg C), + \\
A, + \\
\neg B \lor \neg \neg C, + \\
\neg B \lor C, - \\
D, - \\
\neg B, - \\
C, - \\
\Rightarrow \\
\neg B, + \\
\times \\
\neg \neg C, + \\
\times \\
C, + \\
\times
\end{align*}
\]

The third and fourth lines come from the first, by the rule for true conjunctions. The next line comes from the third by De Morgan’s laws. The next two lines come from the second by the rule for untrue disjunctions, which is then applied again, to get the next two lines. The branching arises because of the rule for true disjunctions, applied to line five. The left
branch is now closed because of \( \neg B, \neg B, + \); an application of double negation then closes the righthand branch.

8.3.7 Here is another example, to show that \( p \land (q \lor \neg q) \not\models r \):

\[
\begin{align*}
p \land (q \lor \neg q), & + \\
r, & - \\
p, & + \\
q \lor \neg q, & + \\
q, & + \quad \neg q, & +
\end{align*}
\]

8.3.8 Counter-models can be read off from open branches in a simple way. For every parameter, \( p \), if there is a node of the form \( p, + \), set \( p_\rho 1 \); if there is a node of the form \( \neg p, + \), set \( p_\rho 0 \). No other facts about \( \rho \) obtain.

8.3.9 Thus, the counter-model defined by the righthand branch of the tableau in 8.3.7 is the interpretation \( \rho \), where \( p_\rho 1 \) and \( q_\rho 0 \) (and no other relations hold). It is easy to check directly that this interpretation makes the premises true and the conclusion untrue.

8.3.10 The tableaux are sound and complete with respect to the semantics. This is proved in 8.7.1–8.7.7.

### 8.4 FDE and Many-valued Logics

8.4.1 Given any formula, \( A \), and any interpretation, \( \rho \), there are four possibilities: \( A \) is true and not also false, \( A \) is false and not also true, \( A \) is true and false, \( A \) is neither true nor false. If we write these possibilities as 1, 0, b and n, respectively, this makes it possible to think of FDE as a 4-valued logic.

8.4.2 The truth conditions of 8.2.6 give the following truth tables:

<table>
<thead>
<tr>
<th>( f_\neg )</th>
<th>( f_\land )</th>
<th>( f_\lor )</th>
<th>( f_\lor )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( f_\neg )</th>
<th>( f_\land )</th>
<th>( f_\lor )</th>
<th>( f_\lor )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>n</td>
<td>1</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>b</td>
<td>n</td>
</tr>
</tbody>
</table>
The details are laborious, but easy enough to check. Thus, suppose that $A$ is $n$ and $B$ is $b$. Then it is not the case that $A$ and $B$ are both true; hence, $A \land B$ is not true. But $B$ is false; hence, $A \land B$ is false. Thus, $A \land B$ is false but not true, 0. Since $B$ is true, $A \lor B$ is true; and since $A$ and $B$ are not both false, $A \lor B$ is not false. Hence, $A \lor B$ is true and not false, 1. The other cases are left as an exercise.

8.4.3 An easy way to remember these values is with the following diagram, the ‘diamond lattice’:

\[
\begin{array}{c}
1 \\
\downarrow & \downarrow \\
\langle & \langle \\
\mid & \mid \\
b & n \\
\uparrow & \uparrow \\
0 & 0
\end{array}
\]

The conjunction of any two elements, $x$ and $y$, is their greatest lower bound, that is, the greatest thing from which one can get to both $x$ and $y$ going up the arrows. Thus, for example, $b \land n = 0$ and $b \land 1 = b$. The disjunction of two elements, $x$ and $y$, is the least upper bound, that is, the least thing from which one can get to both $x$ and $y$ going down the arrows. Thus, for example, $b \lor n = 1$, $b \lor 1 = 1$. Negation toggles 0 and 1, and maps each of $n$ and $b$ to itself.\(^2\)

8.4.4 Since validity in $FDE$ is defined in terms of truth preservation, the set of designated values is $\{1, b\}$ (true only, and both true and false).

8.4.5 This is not one of the many-valued logics that we met in the last chapter, but two of the ones that we did meet there are closely related to $FDE$.

8.4.6 Suppose that we consider an $FDE$ interpretation that satisfies the constraint:

\[
\text{Exclusion: for no } p, p \rho 1 \text{ and } p \rho 0
\]

\(^2\) In fact, this structure is more than a mnemonic. The lattice is one of the most fundamental of a group of structures called ‘De Morgan lattices’, which can be used to give a different semantics for $FDE$. 

i.e., no propositional parameter is both true and false. Then it is not difficult to check that the same holds for every sentence, $A$.\(^3\) That is, nothing takes the value $b$.

8.4.7 The logic defined in terms of truth preservation over all interpretations satisfying this constraint is, in fact, $K_3$. For if we take the above matrices, and ignore the rows and columns for $b$, we get exactly the matrices for $K_3$ (identifying $n$ with $i$). (In $K_3$, $A \supset B$ can be defined as $\neg A \lor B$, as we observed in 7.3.8.)

8.4.8 $K_3$ is sound and complete with respect to the tableaux of the previous section, augmented by one extra closure rule: a branch closes if it contains nodes of the form $A, +$ and $\neg A, +$. (This is proved in 8.7.8.) Here, for example, is a tableau showing that $p \land \neg p \vdash_{K_3} q$. (The tableau is open in $FDE$.)

\[
\begin{array}{c}
p \land \neg p, + \\
q, - \\
p, + \\
\neg p, + \\
\times 
\end{array}
\]

Counter-models are read off from open branches of tableaux in exactly the same way as in $FDE$.

8.4.9 Suppose, on the other hand, that we consider an $FDE$ interpretation that satisfies the constraint:

**Exhaustion:** for all $p$, either $p \rho 1$ or $p \rho 0$

i.e., every propositional parameter is either true or false – and maybe both. Then it is not difficult to check that, again, the same holds for every sentence, $A$.\(^4\) That is, nothing takes the value $n$.

\(^3\) *Proof:* The proof is by an induction over the complexity of sentences. Suppose that it is true for $A$ and $B$; we show that it is true for $\neg A, A \land B$ and $A \lor B$. Suppose that $\neg A \rho 1$ and $\neg A \rho 0$; then $A \rho 0$ and $A \rho 1$, contrary to supposition. Suppose that $A \land B \rho 1$ and $A \land B \rho 0$; then $A \rho 1$ and $B \rho 1$, and either $A \rho 0$ or $B \rho 0$; hence, either $A \rho 1$ and $A \rho 0$, or the same for $B$. Both cases are false, by assumption. The argument for $A \lor B$ is similar.

\(^4\) *Proof:* The proof is by an induction over the complexity of sentences. Suppose that it is true for $A$ and $B$; we show that it is true for $\neg A, A \land B$ and $A \lor B$. Suppose that either $A \rho 1$ or $A \rho 0$; then either $\neg A \rho 0$ or $\neg A \rho 1$. Since $A \rho 1$ or $A \rho 0$, and $B \rho 1$ or $B \rho 0$, then either $A \rho 1$ and $B \rho 1$, and so $A \land B \rho 1$; or $A \rho 0$ or $B \rho 0$, and so $A \land B \rho 0$. The argument for $A \lor B$ is similar.
8.4.10 The logic defined by truth preservation over all interpretations satisfying this constraint is, in fact, $LP$. For if we take the matrices of 8.4.2 and ignore the rows and columns for $n$, we get exactly the matrices for $LP$ (identifying $b$ with $i$). (Again, in $LP$, $A \supset B$ can be defined as $\neg A \lor B$, as we observed in 7.4.6.)

8.4.11 $LP$ is sound and complete with respect to the tableaux of the previous section, augmented by one extra closure rule: a branch closes if it contains nodes of the form $A, -$ and $\neg A, -$. (This is proved in 8.7.9.) Here, for example, is a tableau showing that $p \vdash_{LP} q \lor \neg q$. (The tableau is open in $FDE$.)

$$
p, +  
q \lor \neg q, -  
q, -  
\neg q, -  
\times
$$

Counter-models are read off from open branches of tableaux by employing the following rule: if $p, -$ is not on the branch (and so, in particular, if $p, +$ is), set $p \rho 1$; and if $\neg p, -$ is not on the branch (and so, in particular, if $\neg p, +$ is), set $p \rho 0$.

8.4.12 Finally, and of course, if an interpretation satisfies both Exclusion and Exhaustion, then for every $p$, $p \rho 0$ or $p \rho 1$, but not both, and the same follows for arbitrary $A$. In this case, we have what is, in effect, an interpretation for classical logic. Adding the closure rules for $K_3$ and $LP$ to those of $FDE$, therefore gives us a new tableau procedure for classical logic.

8.4.13 Since all $K_3$ interpretations are $FDE$ interpretations, and all $LP$ interpretations are $FDE$ interpretations, $FDE$ is a sub-logic of $K_3$ and $LP$. It is a proper sub-logic of each, as the tableaux of 8.4.8 and 8.4.11 show.

**8.4a Relational Semantics and Tableaux for $L_3$ and $RM_3$**

8.4a.1 Before we move on to a different kind of semantics for $FDE$, it is worth noting that the semantics for $L_3$ and $RM_3$ can be reformulated in a relational fashion as well. The only difference from $K_3$ and $LP$ (respectively) concerns the appropriate conditional.
8.4a.2 For $L_3$, we consult the truth table of 7.3.8, and recall that $i$ is $n$ — that is, neither true (relates to 1) nor false (relates to 0). It is not difficult to check that:

$$A \supset B \rho 1 \text{ iff } A \rho 0 \text{ or } B \rho 1 \text{ or (none of } A \rho 1, A \rho 0, B \rho 1, \text{ or } B \rho 0)$$
$$A \supset B \rho 0 \text{ iff } A \rho 1 \text{ and } B \rho 0$$

8.4a.3 For $LP$, we consult the truth table of 7.4.6, and recall that $i$ is $b$ — that is, both true (relates to 1) and false (relates to 0). It is not difficult to check that:

$$A \supset B \rho 1 \text{ iff it is not the case that } A \rho 1 \text{ or it is not the case that } B \rho 0 \text{ or (} A \rho 1 \text{ and } A \rho 0 \text{ and } B \rho 1 \text{ and } B \rho 0)$$
$$A \supset B \rho 0 \text{ iff } A \rho 1 \text{ and } B \rho 0$$

8.4a.4 In virtue of these truth conditions, it is straightforward to give tableaux systems for the two logics. The tableaux for $L_3$ are the same as those for $K_3$, with the additional rules for $\supset$:

\[
\begin{array}{c|c}
A \supset B, + & A \supset B, - \\
\hline
\neg A, + & A, \neg B, + \\
B, + & B, - \neg A, - \\
A \vee \neg A, - & \\
\neg (A \supset B), + & \neg (A \supset B), - \\
\hline
A, + & A, - \neg B, - \\
\neg B, + & \\
\end{array}
\]

8.4a.5 The tableaux for $RM_3$ are the same as those for $LP$, with the additional rules for $\supset$:

\[
\begin{array}{c|c}
A \supset B, + & A \supset B, - \\
\hline
A, - \neg B, - & A \land \neg A, + \\
A \land \neg B, + & A, + \neg B, + \\
B, - \neg A, - & \\
\end{array}
\]
8.4a.6 The tableau systems are sound and complete with respect to the appropriate semantics. (See 8.10, problem 11.)

8.5 The Routley Star

8.5.1 We now have two equivalent semantics for FDE, a relational semantics and a many-valued semantics. For reasons to do with later chapters, we should have a third. This is a two-valued possible-world semantics, which treats negation as an intensional operator; that is, as an operator whose truth conditions require reference to worlds other than the world at which truth is being evaluated.

8.5.2 Specifically, we assume that each world, w, comes with a mate, w*, its star world, such that ¬A is true at w if A is false, not at w, but at w*. If w = w* (which may happen), then these conditions just collapse into the classical conditions for negation; but if not, they do not. The star operator is often described with a variety of metaphors; for example, it is sometimes described as a reversal operator; but it is hard to give it and its role in the truth conditions for negation a satisfying intuitive interpretation.

8.5.3 Formally, a Routley interpretation is a structure \( (W, *, \nu) \), where \( W \) is a set of worlds, \( * \) is a function from worlds to worlds such that \( w** = w \), and \( \nu \) assigns each propositional parameter either the value 1 or the value 0 at each world. \( \nu \) is extended to an assignment of truth values for all formulas by the conditions:

\[
\begin{align*}
\nu_w(A \land B) &= 1 \text{ if } \nu_w(A) = 1 \text{ and } \nu_w(B) = 1; \text{ otherwise it is 0.} \\
\nu_w(A \lor B) &= 1 \text{ if } \nu_w(A) = 1 \text{ or } \nu_w(B) = 1; \text{ otherwise it is 0.} \\
\nu_w(\neg A) &= 1 \text{ if } \nu_{w*}(A) = 0; \text{ otherwise it is 0.}
\end{align*}
\]

5 At least, they are equivalent given the standard set-theoretic reasoning employed in the reformulation. Such reasoning employs classical logic, however, and in a set theory based on a paraconsistent logic it may fail. See Priest (1993).