Chapter 3.

Propositional Logic

Section 3.1. Introduction

As we explained in Chapter 1, it is our goal to show how one can represent different domains in some formal system and then to use arguments within this formal system to reach new conclusions. Having done this, we transfer these new conclusions back to the original domain and claim that we have now new information about this original domain. But this means that we must have some method, beyond the formal systems that we use to represent the domains, in which we can carry out these arguments to the new conclusions. This method is called "logic", and can itself be viewed as another formal system. In the formal system of logic, we can explain or define what it is for a result to "follow validly" from previously-found results and what it is for something to be "true in a domain."

Logic is what proofs are defined by, and proofs give us the reasons we believe in the results of mathematics and scientific theories. If, for example, you didn't know that there was a proof of it, why would you believe that the area of a circle was πr^2? The importance of logic in mathematics, science, engineering, etc., cannot be overstated; and it is crucial for you to learn this system well.

In learning logic, one of the things to be learned is how to "translate" statements from the original problem domain into a neutral, domain-independent language (or symbolism). Once the problem is stated in the symbolism, you can then use the "proof methods" of logic to determine what further conclusions can be drawn (or determine whether a proposed conclusion actually does follow) from the initial ones. Now, pretty much *any* problem domain could be chosen to illustrate the method of translating into logic symbolism. But we shall simply content ourselves with showing how to translate from arbitrary natural language (English) statements into our formal logic. (In Chapter 10.2 and 10.4 we consider further translations from the realm of mathematics). The reason for this is that it does not presuppose any previous knowledge of a particular problem domain (e.g., mathematics or some particular science). Furthermore, it is easier to make more "complex" statements in ordinary language than it is in most problem domains, and this will give you practice in some difficult translations. Finally of course, learning this in the most general case of natural language allows the skill to be transferred easily to other domains: generally, if you can translate English statements into logical notation, you can translate statements from any specific problem domain into logical notation.

Section 3.2. Propositions

The logic discussed in this chapter is concerned with ***propositions,*** hence its name "propositional logic." A ***proposition*** is what is expressed by a declarative sentence on some particular occasion of its use. A declarative sentence is a sentence which could possibly be true or false. Here are some examples of declarative sentences:

John went to school today

There are ten planets in our solar system

Force is mass times acceleration

The instructor of this course is extremely clever

Each of these ***could*** be true or ***could*** be false. Of course, whether they ***are*** true or ***are*** false is a matter of possible debate for a variety of reasons. For example, in the first sentence, we need to know which John is being discussed and when "today" is. If you supply this information, then you know which ***proposition*** the sentence expresses; so a proposition is the completely-filled-in thing that is being stated by the declarative sentence. Of course, you still might not know whether it ***is*** true or ***is*** false, but you do know that it is one or the other and not both. Generally, a proposition is what is expressed by a completely definite, declarative sentence. By "completely definite" here we do ***not*** mean that the sentence is about some one specific individual. The second and third sentences are ***general,*** but nonetheless are "completely definite" (except for the use of ***our*** in the second). Rather, being "completely definite" is a matter of expressing a "complete thought".

Not all sentences are declarative sentences. Questions, commands, requests, etc., are not declarative, as in the following

Do you know the answer to problem number 3?

Do your homework!

Pass the salt.

Also, there are various "paradoxical" declarative sentences which do not express propositions, such as

This statement is false.

This cannot be either true or false, since if it were true, then what it says (namely, that it is false) would have to happen and so it would be false; and if it were false then what it says would not happen and so it would be true.

Having discussed what a proposition is, we move on to a discussion of how to represent English in propositional logic. After that, we will discuss some ways of evaluating the truth of expressions in propositional logic (truth tables and various "shortcut" methods). This is followed by a discussion of Conjunctive and Disjunctive Normal Forms for expressions of propositional logic. We then take a brief look at the relationship between the Syntax and the Semantics of propositional logic. (A more detailed discussion is in Chapter 9). Finally, we present an explicit formal system of natural deduction for propositional logic. In Chapter 4, we extend our logic to a system called Predicate Logic.

Section 3.3. Translation

3.3.1. Stylistic Variance

The overall idea of translation is to represent ordinary language (English, here) in a more abbreviated manner, and so that this abbreviated manner exhibits the true *logical form* of the English. You might well ask "Why bother?" The answer to this is not just that the abbreviated way is shorter, but more importantly that the way it is represented will eliminate various infelicities and imperfections of English, and will allow us to draw, more accurately, conclusions when we are given a set of data. As we shall see, since ordinary language is used for a variety of reasons besides the one of drawing conclusions based on information at hand (it is also used for literary effect and so on), there are normally many different ways of saying the same thing. Whenever this happens in ordinary English, we shall want to be able to recognize it and translate both of these alternate ways of saying it into the same abbreviated way. This will be called the *logical form* of each of the sentences, and will be what's important in determining what conclusions can be drawn on the basis of those English sentences. This ability of English to "say the same thing in different ways" is called *stylistic variance,* and the different ways are called *stylistic variants* of each other. Another problem with ordinary English is that sentences which have radically different logical forms sometimes in English have the same grammatical form. For example, the two sentences

John is outside

Nothing is outside

are grammatically both Subject-Copula-Adjective, yet it is obvious that the former sentence says of the object named 'John' that *he* is outside, whereas the second does *not* say anything about an object named 'nothing' -- there is no such object! As we shall later see, we can give these sentences different logical forms by translating them very differently.

In writing computer programs, one often starts by stating an algorithm in ordinary English, then converting this into pseudo-code (or flowchart form), and then finally converting this into explicit Pascal (or FORTRAN, etc.). Our strategy in translating ordinary English into propositional logic will be similar: we will first alter the ordinary English so that it is stated in a "stilted English", and from this stilted English we shall convert it to propositional logic. In the first step the main goal will be to eliminate stylistic variants in favour of one special form. Once we have this special form, the second step will be very easy -- just as in the programming case. This intermediate language, the "stilted English", will be recognizable as English, but will sound peculiar because of the insistence on using only one English form for each type of logical form (rather than ordinary English's ability of making it "sound pretty" by using different stylistic variants).

3.3.2. Simple vs. Truth-Functionally Compound Sentences

A crucial distinction should be made at the first between truth-functionally simple sentences ("atomic sentences") and truth functionally compound sentences. Intuitively speaking, an atomic sentence is a sentence which is either *very* simple in that it does not contain (as a part) any further sentence, or, if it does contain another sentence as a part, it is not a truth-functional part. To explain this, we'll appeal to some examples and then try to give a more detailed account. Here are some truth-functionally *compound* (non-atomic) sentences.

1. Bill sings and Mary dances

2. Either Bill sings or Mary dances
3. If either John or Mary dances, then Bill sings
4. Snow is not white

In these sentences we can see that there are simpler parts which are themselves sentences. For example, in the first sentence, *Bill sings* is such a part, as is *Mary dances*. In the third sentence, *either John or Mary dances* is a part; and this part itself has sentences as parts (with a bit of ellipsis), namely *John dances* and *Mary dances*. In the last one, the sentence which is a part is *Snow is white*. As we mentioned, it is not enough to just have sentences as parts, but also they have to be *truth-functional* parts. This means that *the truth or falsity of the parts entirely determine the truth or falsity of the compound*. For example, the truth or falsity of #1 relies entirely upon whether *Bill sings* and *Mary dances* are each individually true or false. And the truth or falsity of #3 depends entirely upon whether *either John or Mary dances* and *Bill sings* are true or false. (And *either John or Mary dances* itself depends for its truth or falsity entirely upon the truth or falsity of *John dances* and *Mary dances).* The truth or falsity of #4 depends entirely upon the truth or falsity of *snow is white*. Thus these sentences satisfy the definition of being a truth functional compound sentence: they contain sentences as parts and the truth or falsity of the compound depends entirely upon the truth or falsity of these parts. This is in contrast to such complex sentences as

John believes that computing is easy

Kim hit Leslie because Sandy hates spinach

In these sentences there are sub-sentences *(computing is easy* for the first, *Kim hit Leslie* and *Sandy hates spinach* for the second), but the truth or falsity of these sub-sentences is not relevant to the truth or the entire sentence. It is just not relevant, in assessing the truth of the first sentence, whether computing is in fact easy. John might believe it regardless of whether it is true or false. And, so far as the second sentence goes, whether or not it is true that Sandy hates spinach does not tell us *why* Kim hit Leslie. What we say in such cases is that the sentences, while complex, are *not* truth functional compounds of their parts. For, more than just the truth or falsity of the parts is relevant to determining the truth or falsity of the whole.

3.3.3. Logical Connectives and their Stylistic Variants

Sentences that are complex -- whether truth functionally complex or otherwise -- are made up of simpler sentences. These simpler sentences are joined together by *sentence connectives* to make the complex sentence. When the result is a truth-functionally complex sentence, we say that the connectives involved are truth functional connectives. In the above examples, we used *and, or, unless, not,* etc., as truth functional connectives; and we used *believes that* and *because* as non-truth-functional connectives. (Other examples of non-truth-functional connectives are *before, in order that, it is necessary that,* and most "psychological verbs" such as *believes*.)

An atomic sentence is one that is not truth-functionally compound, although it may be complex in a non-truth-functional way. Some examples of atomic sentences are:

Snow is white

There is a blackboard in this room

John got tired because his friend forced him to help paint his house in order to make it more marketable so that he could easily sell it before he moves to Disneyland

It is common to pick out five truth-functional connectives as standard. They are: *it is not the case that, and, or, if-then, if and only if.* However, sometimes even these connectives may be used in a non-truth-functional way, and it is necessary to exercise judgement in deciding whether a particular use of one of these connectives is or is not a truth-functional use. Of course, as we mentioned before, English allows us many stylistic variants of these connectives. Here are some common stylistic variants of our five standard connectives:

it is not the case that X: translated $\sim X$

- it is false that X;
- not-X;
- X is false;
- X is not true;
- X doesn't happen;
- X fails.

X and Y: translated $(X \wedge Y)$

- both X and Y;
- X but Y;
- X although Y;
- X, however Y;
- X, whereas Y;
- X, and also Y;
- X besides Y;
- X, nevertheless Y;
- X, nonetheless Y;
- X even though Y;
- not only X but also Y;
- X in spite of the fact that Y;
- X, but even so, Y;
- X plus the fact that Y;
- X inasmuch as Y;
- X, while Y;
- X, since Y;
- X, as Y;
- X together with Y;
- X as well as Y;
- the conjunction of X and Y.

X or Y: translated $(X \vee Y)$

- either X or Y;
- X or else Y;
- X or, alternatively Y;
- X otherwise Y;
- X with the alternative that Y;
- X unless Y.

if X, then Y: translated $(X \longrightarrow Y)$

- if X, Y;
- Y, if X;
- given that X, (it follows that) Y;
- in case X, Y;
- insofar as X, Y;
- X leads to Y;
- whenever X, Y;
- X only if Y;
- only if Y, X;
- provided that X, Y;
- Y, provided that X;
- so long as X, Y;
- X is a sufficient condition for Y;
- Y is a necessary condition for X.

X if and only if Y: translated $(X \longleftrightarrow Y)$

- X exactly in case Y;
- X just in case Y;
- X when and only when Y;
- X is equivalent to Y;
- X is a necessary and sufficient condition for Y.

(We shall not now go into a long discussion of these translations, but a few things might be noted. First, you should be aware of some terminology. The ***and*** connective is called the "conjunctive operator", it forms a ***conjunction*** and each of its two parts is called a ***conjunct***. The ***or*** connective is called the "disjunctive operator", it forms a ***disjunction*** and each of its two parts is called a ***disjunct***. The ***if-then*** connective is called the "conditional operator", it forms a ***conditional***.[1] The part inside the "if" clause is called the ***antecedent*** of the conditional, and the part inside the "then" clause is called the ***consequent*** of the conditional. The ***if and only if*** connective is called the "biconditional operator". Second, you should especially notice that the conditional operator has a number of pecularities. In English, it can be said in either order: you can say "if X, then Y" or you can say "Y, if X". Furthermore, "if" has a number of stylistic variants such as "provided that", "given that", "in case", "whenever", and the like. So we can combine the two of these and say such things as "Y, whenever X" (which would mean the same as "if X, then Y"). You should also pay particular attention to the fact that ***only***, when it is combined with some stylistic variant of "if", converts the antecedent clause to a consequent clause. Therefore, while "Y, if X" means the same as "if X then Y", the sentence "Y only if X" means the same as "if Y then X". As it turns out, "only" can be combined with almost all the stylistic variants of "if"; so if we were to say "Y only provided that X" we would mean "if Y then X", but if we said "Y provided that X" we would mean "if X then Y". The biconditional is obviously a combination of "if" and "only if" conjoined with "and". Therefore you can get a stylistic variant of "if and only if" by first getting a stylistic variant of "if", then getting a stylistic variant of "only if", and then conjoining them with some stylistic variant of "and". For example, "X if and only if Y" could be stated as "X whenever

[1] Sometimes it is called the "implication operator", forming an ***implication***.

but only whenever Y", or as "X is a necessary and sufficient condition for Y". The *or* that we are interested in is the so-called *inclusive 'or'*. When a statement is formed using this connective, it means that *at least one* of the two disjuncts is true, possibly both. When we have a sentence using *unless* people's intuitions about its meaning differ. We have given it as a stylistic variant of *or*, since that is the simplest way to translate it. Other people, when faced with a sentence like *X unless Y*, prefer to view it as meaning "if X doesn't happen, then Y will"; and still other people prefer to view it as "if Y doesn't happen then X will". These last two viewpoints would translate it respectively as $(\sim X \longrightarrow Y)$ and $(\sim Y \longrightarrow X)$. As we shall later see, these are equivalent to one another, and both equivalent to $(X \vee Y)$. So it really doesn't matter which way one chooses to translate this.)

3.3.4. Atomic Sentences

A truth functional connective (or logical connective, as it is often called; we shall just use the simple "connective" if no ambiguity will result) connects simpler sentences together to make more complex ones. (Although it's a peculiar notion of "connects", since a negation only has one subsentence. It is called a *unary* connective for this reason; the others are *binary* connectives.) Of course these simpler sentences aren't necessarily atomic sentences; the example above of

If either John or Mary dances, then Bill sings

has *either John or Mary dances* as a non-atomic simpler sentence.

We will use upper case letters to stand for or abbreviate atomic sentences of English. Of course, if you had a *lot* of different sentences of English to translate into the formal system, you would perhaps need more than these letters. So we allow ourselves the possibility of using subscripts on these letters. A_0 would then be a different proposition from A_1, and both of these would be different from just plain A. It is often common to use letters with some association to the atomic sentences being abbreviated. So *Bill dances* might be abbreviated as B; and *Mary sings* as M. So in this text, we will adopt the convention that capital letters (usually with some association to the English sentence) will be used to abbreviate atomic sentences.

It would be helpful to have some method of talking about *any* sentence, regardless of whether atomic or compound. We shall use lower case letters, especially p, q, r for this purpose. Capital letters therefore abbreviate atomic sentences and the lower case ones are variables which can be replaced by any sentence whatsoever. (In the next chapter we shall discuss another use we have for lower case letters. We hope that by that time enough will be clear so that context really can distinguish when we mean one and when we mean the other.)

Every sentence is ultimately made up of atomic sentences, connectives, and punctuation. We have discussed how atomic sentences are abbreviated; let us now turn to the connectives. With regard to them, we have five classes, the stylistic variants of: *it is not the case, and, or, if-then, if and only if*. Each class is abbreviated by some symbol. We use one standard set, but for typographical convenience one often sees other symbols. Here is a partial list of some common ones.

English	Our Text	Other Notations
it is not the case that p	$\sim p$	$\neg p$, p', $-p$, $\overline{p}$, Np
p and q	$(p \wedge q)$	$(p \& q)$, $(p \cdot q)$, (pq), Kpq
p or q	$(p \vee q)$	$(p + q)$, Apq
if p then q	$(p \longrightarrow q)$	$(p \supset q)$, Cpq
p if and only if q	$(p \longleftrightarrow q)$	$(p \equiv q)$, Epq

As far as intra-sentential punctuation goes, English has commas, semicolons, dashes,

colons, etc. In our symbolic language the only punctuation are parentheses, although in order to increase visibility and readability we often use different styles of parentheses -- brackets, braces. The point of these punctuation marks is to group together the parts of the sentence which go together, so as to avoid ambiguity. For instance, the string of symbols $(p \vee q \wedge r)$ would be ambiguous between $((p \vee q) \wedge r)$ and $(p \vee (q \wedge r))$. In English a similar ambiguity occurs in "Come with your spouse or come alone and have a good time". To resolve the ambiguity in English we place a comma either after "spouse" or after "alone". The parentheses in the symbolic version perform the same function.

3.3.5. The Formulas of Propositional Logic

Let us return to the task of translating English into our symbolism. It would be good to know first what, precisely, the symbolic language is. Here is an inductive (or recursive) definition of *formula of propositional logic*. (The concept of an inductive or recursive definition will be more fully discussed in Chapter 6.)

1.: Any atomic sentence is a formula.

2.: If p is a formula, then $\sim p$ is a formula.

3.: If p and q are both formulas, then so are

$(p \wedge q)$
$(p \vee q)$
$(p \longrightarrow q)$
$(p \longleftrightarrow q)$

Strings of symbols which do not satisfy 1-3 are not formulas by this definition. And it is these formulas which we are interested in using in propositional logic. (Sometimes these legitimate strings are called *well-formed formulas* (of propositional logic), or simply *wff's* (of propositional logic).) Any string of symbols can be tested against this definition to determine whether it is a formula (of the propositional logic). For example

$$((A \wedge B) \longrightarrow ((C \longleftrightarrow B) \vee \sim A))$$

is a formula, following this reasoning

(a) A, B, and C are all atomic sentences and hence are formulas by 1.

(b) Therefore, $(A \wedge B)$ is a formula by 3; $(C \longleftrightarrow B)$ is a formula by 3; and $\sim A$ is a formula by 2.

(c) Since $(C \longleftrightarrow B)$ and $\sim A$ are formulas, so is $((C \longleftrightarrow B) \vee \sim A)$ by 3.

(d) Since $(A \wedge B)$ and $((C \longleftrightarrow B) \vee \sim A)$ are formulas, so is $((A \wedge B) \longrightarrow ((C \longleftrightarrow B) \vee \sim A))$ by 3.

It might be noticed that rule 2 does *not* introduce parentheses, whereas rule 3 *always* does. This sometimes makes the formulas difficult to read because of the proliferation of parentheses. We've already mentioned the possibility of using different styles of parentheses so as to break up the monotony, so that the above formula might be written as $\{[A \wedge B] \longrightarrow ([C \longleftrightarrow B] \vee \sim A)\}$. Another common practice is to define precedence relations amongst the connectives so that there is always a unique way of restoring parentheses. For instance, it is common to use this ordering

highest: $\sim$

middle: $\vee$, $\wedge$

lowest: $\longleftrightarrow$, $\longrightarrow$

(Often further degrees of precedence are introduced, as sometimes $\wedge$ is given a higher precedence than $\vee$, and sometimes $\longrightarrow$ higher than $\longleftrightarrow$. We shall just stick to the simple three-way list shown.) When faced with a string like $\sim A \vee B$, we recognize that $\sim$ has higher precedence than $\vee$, and so parentheses are restored to yield $(\sim A \vee B)$ rather than $\sim(A \vee B)$. A formula like $A \wedge B \longrightarrow C \vee D$ would have its parentheses restored as $((A \wedge B) \longrightarrow (C \vee D))$ rather than any other way. Our formula of above might therefore be written as

$$A \wedge B \longrightarrow (C \longleftrightarrow B) \vee \sim A$$

A final convention which is often used is that when the truth and falsity of the complex sentence is unaffected by the replacement of parentheses then we may omit them. This is most commonly employed with respect to $\wedge$ and $\vee$. As we shall soon see, $((A \wedge B) \wedge C)$ has the same truth conditions as $(A \wedge (B \wedge C))$; so we can represent it as $(A \wedge B \wedge C)$. The same holds true for $(A \vee B \vee C)$. Of course *mixing* $\wedge$ and $\vee$ does make a difference, as we noted above. $(A \wedge (B \vee C))$ does *not* have the same truth conditions or truth table as $((A \wedge B) \vee C)$, so we cannot eliminate the internal parentheses. And the sentence $((A \longrightarrow B) \longrightarrow C)$ has *very* different truth conditions from $(A \longrightarrow (B \longrightarrow C))$ and so their internal parentheses cannot be removed. $(A \longleftrightarrow (B \longleftrightarrow C))$ and $((A \longleftrightarrow B) \longleftrightarrow C)$ do have the same truth conditions, so their internal parentheses could be removed, although most texts do not do this.

Of course, the real problem is with clarity; so often we will keep parentheses even if they are not strictly necessary.

3.3.6. The Process of Translation

We are now ready to attempt the process of translation from English to our symbolic language. The overall idea is as discussed earlier: first convert the ordinary English to our "stilted English" by eliminating stylistic variance and by judicious placement of parentheses, and then convert the "stilted English" into symbols. This second step is essentially trivial, requiring only the construction of a "scheme of abbreviation" (which is a statement of what abbreviation we shall use for each atomic sentence) and replacement of the standard connectives by their symbolic counterparts. It is the first step which is much more difficult.

Let us proceed by giving an example. As you will see, the crucial part of the procedure has to do with discovering the "main connective" of a sentence (or sentence part), and with being able correctly to spot stylistic variants of our "standard connectives." The procedure we shall develop is sometimes called "top-down parsing" of a sentence. Let us start with a few simple examples. Suppose we are to translate the sentence

> If $3^{15}+1$ is an odd number, then either it is a prime number or the product of two odd numbers.

To translate or parse this sentence we shall wish to find its "main connective". Here the comma tells us that what is being asserted is basically an "if-then" statement, so we keep track of this:

> (if $3^{15}+1$ is an odd number, then either it is a prime number or the product of two odd numbers)

We now notice that the antecedent is an atomic sentence and cannot be further parsed, but that the consequent is itself a complex statement. In particular, it is a disjunction of two simpler statements and the 'it' which serves as a common subject is really $3^{15}+1$. So we replace the "either - or" by the simple "or", fill in the real subject of the embedded sentences, and add the parentheses that go with the "or".

> (if $3^{15}+1$ is an odd number, then ($3^{15}+1$ is a prime number or $3^{15}+1$ is the product of two odd numbers))

We now have parsed this down to atomic sentences. We set up a scheme of abbreviation, such as

O: $3^{15}+1$ is an odd number

P: $3^{15}+1$ is a prime number

N: $3^{15}+1$ is the product of two odd numbers

And we are finally in a position to directly translate the sentence, yielding

$$(O \longrightarrow (P \vee N))$$

Now consider the complex sentence

> If John either plays pool or goes swimming, then, provided that Sally goes to class, he will flunk the midterm unless she gives him the notes.

The first step is to locate the "main connective" of the sentence: ask yourself *what* is being asserted here? English punctuation is a valuable (although not infallible) clue here. In this example, it seems pretty clear that what's being claimed is: *if* such-and-such happens, *then* so-and-so will happen", making the "if-then" be the main connective. (The punctuation of a comma before the 'then' helps, but of course there were other commas in the sentence). Having located the main connective, replace it by the "standard name" of that connective (here "if-then" already is its own standard name), and supply the parentheses that go with it. This yields

> (If John either plays pool or goes swimming, then provided that Sally goes to class, he will flunk the midterm unless she gives him the notes)

Repeat this procedure with the parts remaining. In the antecedent (the "if" clause) we have, pretty obviously, an 'or' statement. And equally obviously the two disjuncts are *John plays pool* and *John goes swimming*. This would yield, after we expand the ellipsis and add parentheses,

> (If (John plays pool or John goes swimming) then provided that Sally goes to class, he will flunk the midterm unless she gives him the notes)

Now let's do it to the consequent. We find the main connective, which is 'provided that' and its comma. We recall that this is a stylistic variant of "if-then"; so we replace it by "if-then" and its parentheses. This yields

> (If (John plays pool or John goes swimming) then (if Sally goes to class then he will flunk the midterm unless she gives him the notes))

Turning our attention to the last part of the sentence, we recognize that its main connective is 'unless' and recall that this is a stylistic variant of "or". So we replace it by "or", add its associated parentheses, and fill in the referents of *she* and *he*, yielding

> (if (John plays pool or John goes swimming) then (if Sally goes to class then (John will flunk the midterm or Sally gives John the notes)))

Since there are no more connectives to consider, we are now ready to replace the atomic sentences by their abbreviations and the standard connectives by their abbreviations. So we construct a scheme of abbreviation, such as

P: John plays pool

S: John goes swimming

C: Sally goes to class

F: John will flunk the midterm

G: Sally gives John the notes

and replace, yielding

$$((P \vee S) \longrightarrow (C \longrightarrow (F \vee G)))$$

Deleting a few parentheses, we get

$P \vee S \longrightarrow (C \longrightarrow F \vee G)$

Practice is the only way to become proficient at this. Try it with

a: If it snows or freezes tomorrow, then if the kumquats are in blossom and are unprotected, then the crop will be ruined unless a miracle occurs.

b: Not both Fred and Randy will win the programming contest.

c: Kim wants someone to take him on a date, but Sandy's car isn't working and Leslie is sick in bed.

d: If Len gets sick if he drinks too much, then given that he is healthy if and only if he is sober, he drinks too much if he gets sick.

We shall do it with one more sentence. Recall that *p only if q* is a stylistic variant of *if p then q*. Further we remark that *neither p nor q* is a stylistic variant of both of *not-(either p or q)* and *(not-p and not-q)*. (You can look at *neither...nor...* as a way of negating an entire *either...or...* -- which gives you the former, or as saying that "neither are true", i.e., both are false -- which gives you the latter. As we shall see, these are equivalent ways of saying the same thing.) Consider the sentence

If John goes to school only if Mary neither stays home nor goes to work, then neither one will be happy unless both are crazy.

We first spot "if-then" as the main connective. This yields

(if John goes to school only if Mary neither stays home nor goes to work then neither one will be happy unless both are crazy)

Now, working on the antecedent, we spot "only if" as the main connective, giving us

(if (if John goes to school then Mary neither stays home nor goes to work) then neither one will be happy unless both are crazy)

The *Mary neither stays home nor goes to work* part is a stylistic variant of *it is not the case that (Mary stays home or Mary goes to work)*, so we get

(if (if John goes to school then not- (Mary stays home or Mary goes to work)) then neither one will be happy unless both are crazy)

Turning our attention to the consequent, we recognize *unless* as the main connective (and a stylistic variant of *or)*. So we get

(if (if John goes to school then not- (Mary stays home or Mary goes to work)) then (neither one will be happy or both are crazy))

The *neither one will be happy* becomes *it is not the case that (John will be happy or Mary will be happy)*. This yields

(if (if John goes to school then not- (Mary stays home or Mary goes to work)) then (not- (John will be happy or Mary will be happy) or both are crazy))

And lastly, the *both are crazy* is a stylistic variant of *John is crazy and Mary is crazy* so overall we get

(if (if John goes to school then not- (Mary stays home or Mary goes to work)) then (not- (John is happy or Mary is happy) or (John is crazy and Mary is crazy)))

which is our "stilted English" version of the original sentence. We now construct a scheme of abbreviation for the atomic sentences, such as

S: John goes to school

H: Mary stays home

W: Mary goes to work

J: John will be happy

M: Mary will be happy

C: John is crazy

D: Mary is crazy

Doing all the replacements yields

$((S \longrightarrow \sim(H \vee W)) \longrightarrow (\sim(J \vee M) \vee (C \wedge D)))$

Deleting some parentheses, we can reduce this to

$(S \longrightarrow \sim(H \vee W)) \longrightarrow \sim(J \vee M) \vee (C \wedge D)$

Section 3.4. Truth Tables

Truth tables are a method to tabulate whether a truth-functionally compound sentence is true or false, based on the truth or falsity of the components. If, for example, you know that p is true, then you automatically know $\sim p$ to be false (and the reverse). Of course, you probably don't know whether p *is* true or *is* false, so we give a table consisting of all the possibilities, namely two. Here is a truth table for $\sim p$.

p	$\sim p$
T	F
F	T

The left side of the vertical line says: "when p has the value __" and the right side says "then $\sim p$ has the value __". When we are considering the binary connectives, each of the parts can be true or false independently, so there are four possibilities to be considered. The various binary connectives we have considered have the following truth tables.

p	q	$(p \wedge q)$	$(p \vee q)$	$(p \longrightarrow q)$	$(p \longleftrightarrow q)$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

As you can easily see, there are 16 possible binary truth functions. We have picked these four because of their nearness to various English connectives. (Later on, we shall mention some of the others such as NAND, NOR, and XOR). Given these basic truth tables for the four binary and one unary connectives, we can construct a truth table for any complex sentence by building it up from the truth tables of its parts. Sometimes when the final truth table for a sentence is given, it will consist of all T's. Such sentences are called *tautologies*. Other times the truth table will consist of all F's in its final column. Such sentences are called *contradictions*. Sentences which have truth tables that are not all T's and not all F's are called *contingencies*.

Let us look a bit at the process of writing a truth table for an arbitrary formula, say $(\sim(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$. The strategy will be to first figure out all the possible combinations of T and F for the three sentence letters occurring in our formula. This is easily done by writing a table that consists of a left side and a right side. The left side has columns for each sentence letter. (The right side will be left blank for now).

P	Q	R	

Now, for the column closest to the vertical bar (here, our R), alternate T's and F's.

For the next column (our Q), alternate *pairs* of T's and F's, and for the third column (our P), alternate *quadruples* of T's and F's. This yields

P	Q	R	
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

If there had been a fourth sentence letter, it would have alternated *octuples* of T's and F's. The total number of rows needed to capture all the possible truth combinations for the sentence letters obviously depends on how many sentence letters there are: if there are n sentence letters, we need 2^n rows.

For the second step, we use the right half of the table. We construct a column for *each* subformula of $(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$. Since the left side of the table already contains columns for the atomic sentences, the right side needs to have columns only for the more complex ones. In this example, the more complex ones are $\sim Q$, $(P \longrightarrow Q)$, $(R \vee \sim Q)$, $\neg(P \longrightarrow Q)$, and the entire formula.

P	Q	R	$\sim Q$	$(P \longrightarrow Q)$	$(R \vee \sim Q)$	$\neg(P \longrightarrow Q)$	$(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$
T	T	T					
T	T	F					
T	F	T					
T	F	F					
F	T	T					
F	T	F					
F	F	T					
F	F	F					

Now, the basic truth tables for the connectives have been given above. We can just apply them here to fill in the present table. For example, the basic truth table for $\sim p$ says to change p's value. In the present example, under $\sim Q$ we would enter the opposite of Q's value (which we get from the column under Q). This would yield

P	Q	R	$\sim Q$	$(P \longrightarrow Q)$	$(R \vee \sim Q)$	$\neg(P \longrightarrow Q)$	$(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$
T	T	T	F				
T	T	F	F				
T	F	T	T				
T	F	F	T				
F	T	T	F				
F	T	F	F				
F	F	T	T				
F	F	F	T				

Now we move to the next column, $(P \longrightarrow Q)$, and fill in its truth table (by referring to the basic truth table for $(p \longrightarrow q)$ and using the P and Q columns).

P	Q	R	$\sim Q$	$(P \longrightarrow Q)$	$(R \vee \sim Q)$	$\neg(P \longrightarrow Q)$	$(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$
T	T	T	F	T			
T	T	F	F	T			
T	F	T	T	F			
T	F	F	T	F			
F	T	T	F	T			
F	T	F	F	T			
F	F	T	T	T			
F	F	F	T	T			

Now go to the $(R \vee \sim Q)$ column, using the basic truth table for $(p \vee q)$. Here the relevant subparts are the R column and the $\sim Q$ column.

P	Q	R	$\sim Q$	$(P \longrightarrow Q)$	$(R \vee \sim Q)$	$\neg(P \longrightarrow Q)$	$(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$
T	T	T	F	T	T		
T	T	F	F	T	F		
T	F	T	T	F	T		
T	F	F	T	F	T		
F	T	T	F	T	T		
F	T	F	F	T	F		
F	F	T	T	T	T		
F	F	F	T	T	T		

And now we move to the $\neg(P \longrightarrow Q)$ column. Again we use the basic truth table for negation, and apply it to the $(P \longrightarrow Q)$ column.

P	Q	R	$\sim Q$	$(P \longrightarrow Q)$	$(R \vee \sim Q)$	$\neg(P \longrightarrow Q)$	$(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$
T	T	T	F	T	T	F	
T	T	F	F	T	F	F	
T	F	T	T	F	T	T	
T	F	F	T	F	T	T	
F	T	T	F	T	T	F	
F	T	F	F	T	F	F	
F	F	T	T	T	T	F	
F	F	F	T	T	T	F	

We are finally ready to determine the truth table for our original formula, which is in the far right hand column. We fill in its entries using the basic $(p \longleftrightarrow q)$ truth table and the columns for $\neg(P \longrightarrow Q)$ and $(R \vee \sim Q)$

P	Q	R	$\sim Q$	$(P \longrightarrow Q)$	$(R \vee \sim Q)$	$\neg(P \longrightarrow Q)$	$(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$
T	T	T	F	T	T	F	F
T	T	F	F	T	F	F	T
T	F	T	T	F	T	T	T
T	F	F	T	F	T	T	T
F	T	T	F	T	T	F	F
F	T	F	F	T	F	F	T
F	F	T	T	T	T	F	F
F	F	F	T	T	T	F	F

This final column (together with the specific order you chose on the left side of the vertical bar) is the truth table for $(\neg(P \longrightarrow Q) \longleftrightarrow (R \vee \sim Q))$. We see that it is true in four cases:

a. P and Q are T, R is F

b. P and R are T, Q is F

c. P is T, Q and R are F

d. Q is T, P and R are F

It is false in the other four cases; hence, it is a contingency.

Section 3.5. Equivalences

In the above basic truth tables of the last section, note particularly the column for '$\leftrightarrow$'. This table says that p and q have the same truth value: if p is true then so is q, and if p is false then so is q. Hence if we wish to say that two formulas are *equivalent* we can do it in one of two ways: (1) we could say that they have the same truth table, or (2) we could say that the result of placing a $\leftrightarrow$ between them is a tautology. Either of these ways can be tested by truth tables. (By the way, we can justify our earlier claims in this manner. Write a truth table for each of $(p \wedge (q \wedge r))$ and $((p \wedge q) \wedge r)$. You will discover them to be the same, so we are justified in our practice of dropping the internal parentheses -- it wouldn't matter which way you added them back on. Now write a truth table for $\sim(p \vee q) \leftrightarrow (\sim p \wedge \sim q)$. You will discover that there are all T's in its final column; so it's a tautology. This means that $\sim(p \vee q)$ and $(\sim p \wedge \sim q)$ are equivalent ways of saying the same thing. Recall that we said that *neither p nor q* could be translated either way.)

Knowledge of certain of these equivalents, especially the "DeMorgan Laws" which relate '$\wedge$', '$\vee$', and '$\sim$' can make your programming life much easier. Most programming languages have "connectives" corresponding to these three. For example, Pascal has 'AND', 'OR', and 'NOT'. One type of "atomic expression" in Pascal is simple equality, greater than, and less than between variables. So 'X<Y', 'A=B', and the like are "atomic expressions" which can be either true or false, and which can be made into compound expressions by means of the connectives. Pascal (and other programming languages) use such expressions to control the action of a loop. Two loop structures in Pascal are

```
WHILE p DO
      <body>
```

and

```
REPEAT <body>
      UNTIL p
```

The "while loop" works as follows: the statement(s) in the "body" are continually performed so long as p is true. p is checked for truth or falsity, and if it is true then the "body" is performed and p is again checked. If it is true the process is repeated. When p is false the "body" is not performed, and control is passed to the next statement in line. The "until loop" performs the "body" until p becomes true -- that is, as long as p is false. (Actually, it performs the body, then checks for p's truth or falsity. If p is false, then it does it all again. If p is true, control is passed to the next statement.)

Suppose you wish to perform some action as long as some complex state of affairs is true. Let's say you want to perform "body" as long as either A<B or B=50. If you were to use the while loop you would say

```
WHILE ((A<B) OR (B=50)) DO
      <body>
```

If you were to use an until loop you would say

```
REPEAT <body>
      UNTIL NOT((A<B) OR (B=50))
```

Using the DeMorgan Laws you could alternatively put this last loop as

```
REPEAT <body>
      UNTIL (NOT(A<B) AND NOT(B=50))
```

But these are easy cases. Suppose instead you want a loop to stop when either A>B or A<C, and you were to use a while loop. You should say to yourself: "The loop is to stop when one or the other of these statements becomes true. That is, it continues so long as the disjunction is false." So you write

```
WHILE NOT((A>B) OR (A<C)) DO
      <body>
```

You might also appeal to DeMorgan's Laws and say to yourself: "So it stops when one or the other of the statements becomes true. So it must continue as long as they're both false." So you might write

```
WHILE (NOT(A>B) AND NOT(A<C)) DO
      <body>
```

Suppose the desire was instead to write a while loop to stop when both A=B and B=C. You say "So it is to stop when (A=B AND B=C); it must therefore continue as long as this compound statement is false." So you write

```
WHILE NOT((A=B) AND (B=C)) DO
      <body>
```

Or, using DeMorgan's Laws, you write

```
WHILE (NOT(A=B) OR NOT(B=C)) DO
      <body>
```

Finally suppose you want to loop to stop when neither A=B nor B=C. Again you say: "Stops when NOT(A=B OR B=C). Therefore continues so long as this compound statement is false." So you write

```
WHILE ((A=B) OR (B=C)) DO
      <body>
```

A good grasp of these simple equivalences helps programming a lot. If you don't know the answer off the top of your head, you can always write a truth table to figure it out.

Section 3.6. Truth Table Shortcuts and Related Methods

3.6.1. Some Shortcuts

As mentioned above, to consider every possible assignment of T or F to each of the n sentence letters of a given complex sentence would require a truth table with 2^n rows in it. When there are more than three different sentence letters in a sentence, the construction of an entire truth table becomes extremely long and tedious. Therefore various shortcuts have been developed to aid in evaluating such sentences. Using the shortcuts depends on what you wish to show about the sentence in question. For example, if you wonder whether the formula is a tautology, you might check only those rows which you don't already know that it's true. Consider, for example, a sentence like $(p \wedge q) \longrightarrow p$. We know from the '$\longrightarrow$' truth table that the only time an '$\longrightarrow$' statement can be false is when its antecedent is true. And since the antecedent