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 $(X \rightarrow Y)$
- 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 $(X \leftrightarrow Y)$
- 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

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

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

$((S \rightarrow \neg(H \vee W)) \rightarrow (\neg(J \vee M) \vee (C \wedge D)))$

Deleting some parentheses, we can reduce this to

$(S \rightarrow \neg(H \vee W)) \rightarrow \neg(J \vee M) \vee (C \wedge D)$

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 $\neg p$ 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 $\neg p$.

p	$\neg p$
T	F
F	T

The left side of the vertical line says: "when p has the value __" and the right side says "then $\neg p$ 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	$(p \wedge q)$	$(p \vee q)$	$(p \rightarrow q)$	$(p \leftrightarrow q)$
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 $(\neg(P \rightarrow Q) \leftrightarrow (R \vee \neg Q))$. 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 $(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$. 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 $\sim Q$, $(P\longrightarrow Q)$, $(R\vee\sim Q)$, $\sim(P\longrightarrow Q)$, and the entire formula.

P	Q	R	$\sim Q$	$(P\longrightarrow Q)$	$(R\vee\sim Q)$	$\sim(P\longrightarrow Q)$	$(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$
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	$\sim Q$	$(P\longrightarrow Q)$	$(R\vee\sim Q)$	$\sim(P\longrightarrow Q)$	$(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$
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	$\sim Q$	$(P\longrightarrow Q)$	$(R\vee\sim Q)$	$\sim(P\longrightarrow Q)$	$(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$
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	$\sim Q$	$(P\longrightarrow Q)$	$(R\vee\sim Q)$	$\sim(P\longrightarrow Q)$	$(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$
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 $\sim(P\longrightarrow Q)$ column. Again we use the basic truth table for negation, and apply it to the $(P\longrightarrow Q)$ column.

P	Q	R	$\sim Q$	$(P\longrightarrow Q)$	$(R\vee\sim Q)$	$\sim(P\longrightarrow Q)$	$(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$
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 $\sim(P\longrightarrow Q)$ and $(R\vee\sim Q)$

P	Q	R	$\sim Q$	$(P\longrightarrow Q)$	$(R\vee\sim Q)$	$\sim(P\longrightarrow Q)$	$(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$
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 $(\sim(P\longrightarrow Q)\longleftrightarrow(R\vee\sim Q))$. 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

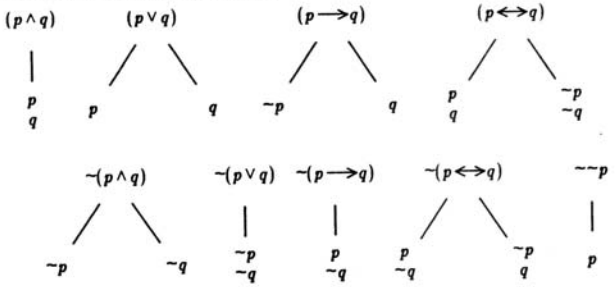
3.6.1. Some Shortcuts

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

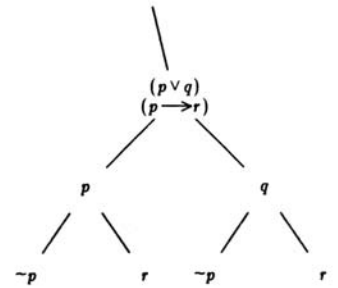
is $(p \wedge q)$ and the only time a conjunction is true is when both conjuncts are true, this is the only case we have to look at. In other words, we need only look at the first row of the truth table. As we discover when looking at this row, the consequent, p, is also true. (And therefore, according to the '$\longrightarrow$' truth table, the entire sentence is true in this row). But as we said, we already knew that this was the only possible case where the formula might have been false, and so it must be a tautology because even here it is true. Alternatively, we might have said to ourselves that the only time this sentence could have been false was when the consequent was false. Therefore the only lines of the truth table we need look at are the last two (where p is false). But then we notice that in these two rows, $(p \wedge q)$ is false, so that the entire conditional must be true. So, you say to yourself, even in these rows the sentence is true, so it must be a tautology.

3.6.2. Truth Trees

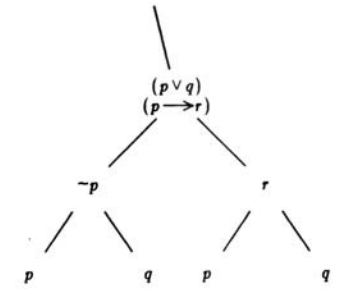
Another method often used in evaluating sentences for tautologousness (and, as we shall see later, for evaluating the validity of arguments) is called "truth trees". In a wide variety of cases the truth tree method (which we will state as an algorithm) can quickly yield an answer to whether a formula is a tautology, and if not, what rows of the truth table create the F's which prevent the formula from being a tautology. Before giving the algorithm, let us introduce "branching rules". There are nine such rules: one for each binary connective and one for the negation of any connective. The idea is that we shall construct a "tree" (don't worry about what precisely a tree is -- the idea will be clear enough) to test the formula we're interested in. Depending on just how the formula is constructed by the connectives, we shall "break it down" using the branching rules, thereby constructing a diagram (a "tree"). The nine branching rules tell us how to break down any complex formula (except simple negations) into its parts by making the tree bigger. Of course, when you "break down" a formula you might discover that one of its parts is itself complex, and so you have to "break it down" by using the branching rule appropriate to it. It is the continual "breakup" of complex formulas which constructs the truth tree. Every complex formula can be "broken down" by the branching rules.



As you can see, some of the rules introduce more than one formula on a node of the tree. Each of these formulas is eligible for being broken up, although you operate on just one at a time. This means we can be led to such sequences of "breakdowns" as the following. We might have a node of the tree with two formulas on it, and "break up" one of them (the order of breakdown never matters, except for efficiency), yielding two different branches. Now, to break up the other formula, we have to break it up on *both* branches, like this



Here we branched the $(p \vee q)$ to yield two branches, and then branched the $(p \longrightarrow r)$ to yield two branches under each of the former branches. We could have done it the other way: branch $(p \longrightarrow r)$ first to yield two branches and then branch $(p \vee q)$. But here too we must branch the second formula under both of the branches we got from $(p \longrightarrow r)$.



You should note -- we will mention why it is important later -- that the branching rules all have the property that: the formula being branched is true if and only if every formula on at least one branch is true. For instance, $(A \vee B)$ yields an A branch and a B branch. The $(A \vee B)$ is true just in case at least one of A and B is true. $\neg(A \longrightarrow B)$ yields only one branch, which contains both A and $\neg B$. $\neg(A \longrightarrow B)$ is true (that is, $(A \longrightarrow B)$ is false) exactly in case A and $\neg B$ are both true (that is, A is true and B is false). The same is true for all our branching rules.

We are now ready to state the tautology-testing algorithm using truth trees. The algorithm for evaluating a formula says to do the following.

(1) start by writing *the negation* of the formula to be tested as the root of the tree. (The negation of the *entire* formula).

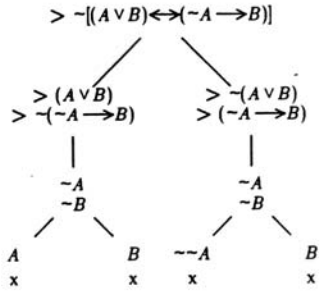
(2) Apply a branching rule to any unchecked formula in the tree, writing the result of the branching under every "active" (to be defined soon) path beneath the formula,

(3) check off (with a '>') the formula you just used the branching rule on,

(4) if any path from the root of the tree to a leaf contains a formula and also the negation of that formula, put an 'x' under that path. Any path with an 'x' under it is not active and need be considered no further. All other paths are active.

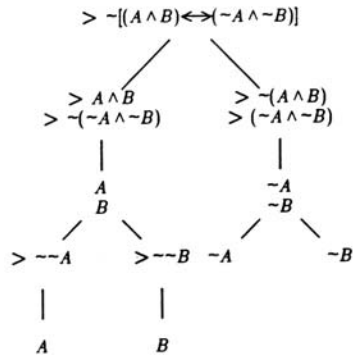
(5) Repeat steps (2)-(4) until either (a) there are no more active paths, or (b) there is an active path but there are no more unchecked complex formulas on that path.

If you reach step (5a), then the original *unnegated* formula was a tautology. If you reach step (5b) then the original *unnegated* formula was not a tautology and every open path describes some row (or rows) of the truth table where it was false. Let us consider two examples, the first of a tautology and the second of a contingency. The first formula is $(A \vee B) \leftrightarrow (\neg A \rightarrow B)$



Since no path is active, the original formula, $(A \vee B) \leftrightarrow (\neg A \rightarrow B)$ is a tautology.

The second formula, which is not a tautology, is $(A \wedge B) \leftrightarrow (\neg A \wedge \neg B)$



Here every path is active. (The fact that *all* paths are active is irrelevant, all that's important is that *some* path is still active.) Each open path describes some row of the truth table according to which $[(A \wedge B) \leftrightarrow (\neg A \wedge \neg B)]$ is false. We can describe the relevant rows by looking at a path and seeing which atomic sentences occur in it unnegated and which occur negated. In the leftmost branch, for example, both A and B occur. The relevant row is where both A and B are true. The second-to-left path also has both A and B in it and therefore describes the same row of the truth table. The third and fourth paths both have $\neg A$ and $\neg B$ in them, and the relevant row of the truth table is where both A and B are false. (When checking an active path, it may happen that some sentence letter does not occur in it. In this case, we have described two rows of the truth table, one where the missing letter is T and one where it is F. If two letters are missing from a path, then we have described four rows of the truth table, etc.) These rows of the truth table which show that the formula is not a tautology are called *counterexamples* (to the formula).

This tree method gives a rather quick way of checking truth tables, especially in those cases where there are a lot of sentence letters. The method works because of the interplay of two things. First, the branching rules have the properties that, if the branched sentence is true then so is at least one of the branches, and conversely. This means that every open path describes a way that all the sentences on that path might be true, including the sentence at the root. And since these are *all* the ways a sentence might be true, it follows that there are no rows of truth tables which have been ignored. Second, we have placed the negation of the sentence at the root. So if all paths are non-active, there is no way this negation can be true; and hence the unnegated sentence must be a tautology. If some path is still active, then it *does* describe a way the root (negated sentence) can be true, and hence a way the unnegated sentence can be false.

Section 3.7. Arguments

An *argument* is a set of sentences in which one of the sentences (called the *conclusion*) is claimed to "follow from" the others (called the *premises*). For example, the following collection of sentences forms an argument.

> Either it is not the case that Leslie pays attention and does not lose track of the argument, or it is not the case that she does not take notes and does not do well in the course. Leslie neither does well in the course nor loses track of the argument. If Leslie studies logic, then she does not do well in the course only if she does not take notes and pays attention. Therefore Leslie does not study logic.

You can tell this is an argument in part by the context -- here someone is trying to convince you of something (that Leslie does not study logic) by adducing some reasons. Also the fact that the word *therefore* occurs here gives the information that you are supposed to become convinced. Such words as *therefore, hence, so, it follows that,* and the like, function as conclusion-indicators. They tell you that the sentence following is the conclusion of the argument. (Not all arguments in English will necessarily have the conclusion at the end). Words like *because, for the reason that, on account of,* and the like, are premise-indicators telling you that the associated sentences function as premises to the argument.

Virtue, in an argument, is called *validity*. An argument is valid if and only if the conclusion really *does* "follow from" its premises. Conversely, a bad argument is called *invalid*. So the question arises: how can we determine whether an argument is good or bad, virtuous or unvirtuous? There are, generally speaking, two ways that can be used. One way is to construct an explicit formal proof of the translation into symbolic form of the argument. This method will be discussed later in this chapter. The other way is to use one of the truth table methods. This way rests upon understanding what "follows from" means in terms of the truth or falsity of the parts of an argument.

Definition 3.7.1: valid argument
An argument is (truth functionally) *valid* if and only if: In any row of a truth table in which all the premises are true, so is the conclusion.

To determine whether an argument is valid according to this definition, you need to write a truth table (or use some other truth method). If you wrote a truth table it would contain columns for *every* atomic sentence letter that occurred anywhere in the argument. This typically means that the truth table would be very large. If this method is employed, you write a column for each premise and for the conclusion. Once their truth tables have been constructed it is a simple matter to see whether there is any row in which all the premises are true and the conclusion false. If there is such a row, then the argument is invalid, otherwise it is valid. If it is invalid, then the row found describes (one) *counterexample:* the T's and F's assigned in that row to the atomic sentences tells you what state of affairs could happen in the world that would make all the premises be true and the conclusion false. That is, it shows you why the argument is invalid.

Of course not every argument in ordinary language is so carefully stated as the one above. Often, certain premises are left out especially when the speaker views them as "obvious". According to the definition above, such enthymatic arguments (as they are called) are invalid. But in another sense they are good arguments and could be stated validly merely by making explicit these unstated, "obvious" premises. Finding the literal counterexample can often help in uncovering precisely what the "hidden" premises are.

Let us symbolize the argument above. We use the following scheme of abbreviation.

P: Leslie pays attention
L: Leslie loses track of the argument
N: Leslie takes notes
W: Leslie does well in the course
S: Leslie studies logic

And the argument is symbolized as (can you get the same answer?):

$\sim(P \wedge \sim L) \vee \sim(\sim N \wedge \sim W)$
$\sim(W \vee L)$
$S \rightarrow (\sim W \rightarrow (\sim N \wedge P))$
$\therefore \sim S$

We note that there are five atomic sentences here, so a complete truth table would have 2^5, i.e., 32, lines in it. It could be written, but it would be tedious. We could also try one of the shortcut methods. Since we wish to know whether it is valid, we are interested in whether it is possible that all premises are true and the conclusion false. (If this *is* possible, then the argument is invalid.) To try to find this out, we might start by making $\sim S$ be false, that is we start by assigning

S: T

The second premise is supposed to be true (as are all premises), so, since it is a negation, the internal disjunction should be false, therefore

S: T
W: F
L: F

If the third premise is to be true, then since S is true, $\sim W \rightarrow (\sim N \wedge P)$ must also be true. But since $\sim W$ is already required to be true, it follows that $(\sim N \wedge P)$ must be true, i.e.,

S: T
W: F
L: F
N: F
P: T

Given all this, let's see if we have succeeded in making the first premise true. If we have, the argument is invalid. If we haven't, then since we were forced to use these assignments to make premises two and three true and the conclusion false, the argument must be valid. (Because there is no possible way to make all premises true and the conclusion false). We could write a truth table for premise one to see whether it is true or false in the row described by the already-given assignment of truth values to the atomic sentences. But we could also do it in a quicker manner. Since L is false, $\sim L$ is true; and since P is true, it follows that $(P \wedge \sim L)$ is true. So, the first part of premise 1, $\sim(P \wedge \sim L)$, is false. Therefore, if premise one is to be true, the second disjunct must be true. But since both W and N are false, $(\sim N \wedge \sim W)$ must be true and so $\sim(\sim N \wedge \sim W)$ must be false. Therefore, we cannot make all premises true and the conclusion false, and so this is a valid argument.

Let's retest this argument by truth trees. Recall that the account of truth trees given earlier tested an individual sentence for being or not being a tautology: We negated the sentence and checked whether there were any active paths left in the tree. If not, the sentence was a tautology (since there was no possible way for the negation of the sentence to be false). If so, it wasn't and we could find at least one counterexample. With a minor modification of the algorithm given there (the only

change is in step 1) we can apply that method to arguments.

1. At the root of the tree, list all premises and the *negation* of the conclusion.

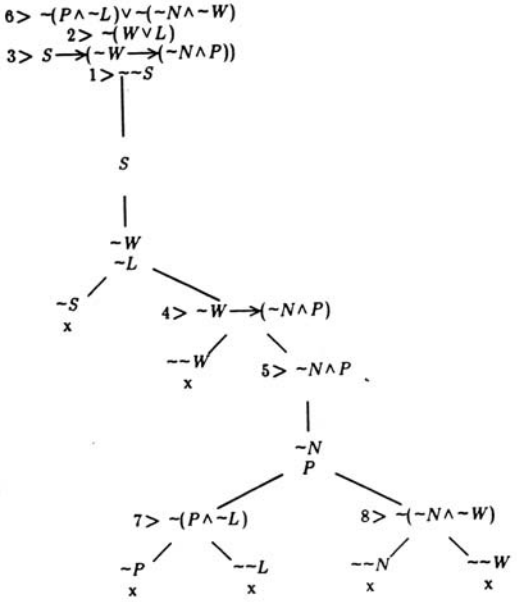
2. Apply a branching rule to any unchecked formula in the tree, writing the result under every active path beneath the formula.

3. Check off the formula you just branched.

4. If any path from the root of the tree to a leaf contains a formula and also the negation of that formula, put an 'x' under that leaf. Any path with an 'x' under it is not active.

5. Repeat steps (2)-(4) until either (a) there are no more active paths, or (b) there is an active path but no more unchecked complex formulas.

If you reach step (5a) then the argument is valid, because it is not possible to have all the premises and also the negation of the conclusion all be true. If you reach step (5b), then it *is* possible, and you can use the method described earlier to find the counterexample(s) to the argument. Here is the truth tree method applied to the above argument. (The numbers beside the "checks" merely give the order in which we applied the branching rules).



Since all paths close, the argument is valid.

Section 3.8. Normal Forms

A *normal form* of a formula is a (possibly different) formula which has the same truth table as the original formula and furthermore is written in a certain way. For example, we might want our formulas not to have any $\leftrightarrow$ in them. We could replace any part of a formula that does have a $\leftrightarrow$ in it by a truth table equivalent representation, for instance replace any part like $(p \leftrightarrow q)$ with $((p \rightarrow q) \wedge (q \rightarrow p))$. This would give us a formula which we might call the biconditional-free normal form. This is not a particularly interesting type of normal form, but other types are more interesting.

In working with logical formulas, it is often convenient to restrict our attention to formulas of some particular simple form. In this section, we will define several types of "normal forms" for formulas, and we will show how to convert any formula into normal form.

Objective 3.8.1
Give reasons for studying normal forms.

Objective 3.8.2
Define disjunctive normal form and conjunctive normal form.

Objective 3.8.3
Show how to convert any propositional formula into disjunctive or conjunctive normal form.

Objective 3.8.4
Illustrate some applications of normal forms.

Before we can begin to talk about normal forms, we must specify precisely what it means for two formulas to be "equivalent". Once we have introduced the appropriate definitions and notation for equivalence, the definitions of normal forms will follow quite naturally.

Definition 3.8.1: logical equivalence
Two formulas are *logically equivalent* (or just *equivalent*, for short) if they represent the same propositional function.

In other words, two formulas (or complex propositions) are equivalent if they have the same truth table. (Strictly speaking, this is not true. The formulas q and $((p \rightarrow q) \wedge (\sim p \rightarrow q))$ are equivalent, since they represent the same propositional function, but their truth tables have different numbers of rows and columns. Note that the truth tables are the same, however, in the sense that for every row with a T in the p column there is an otherwise identical row with an F in the p column. When the p column is deleted, and the identical rows are merged, the two truth tables become identical.)

Notation: $=$
If p and q are equivalent formulas, we write $p = q$.

In order to be able to talk about propositional formulas in general, we give a special name to the set of all formulas.

Definition 3.8.2: F
$\mathbf{F} = \{p \mid p \text{ is a propositional formula}\}$.

Then $=$ is simply a relation on $\mathbf{F}$. In fact, $=$ is an equivalence relation. (We go into this more in Chapter 6. You might look at this now and try to think of what you have to show in order to prove this fact!) This justifies our use of the term "equivalent" for $=$.

Practice 3.8.1
Prove that $\equiv$ is an equivalence relation on **F**. [Keep this problem in mind for later in the text.]

Definition 3.8.3: equivalence class
The set of formulas that are equivalent to some formula p is called the *equivalence class* of p.

Notation: [ ]
The equivalence class of p is denoted by $[p]$.

That is, $[p] = \{q \mid q \in \mathbf{F} \text{ and } q \equiv p\}$, or in words, $[p]$ is the collection of all of those formulas that have the same truth table as p. Then, in terms of understanding or proving anything about the propositional function represented by p, we could just as well use any other formula q in $[p]$, since q represents the same propositional function as p does. Depending on what it is we want to do or know, some formulas in $[p]$ may be easier to work with than others.

The idea behind normal forms is to define a subset **N** of formulas that are "easy to work with" in some sense, such that every formula is equivalent to some formula in **N**. This idea leads to the following definition.

Definition 3.8.4: normal form
If $\mathbf{N} \subseteq \mathbf{F}$, then **N** is a *normal form* if, for every formula $p \in \mathbf{F}$, there is at least one formula q such that $q \in \mathbf{N}$ and $q \in [p]$ (*i.e.* $q \equiv p$).

The definition of normal forms often makes things simpler by allowing us to restrict our attention to formulas in **N**. The most common normal forms are defined by allowing only certain kinds of syntactically simple formulas. We will consider two such examples: disjunctive normal form, and conjunctive normal form.

Definition 3.8.5: literal
A *literal* is either an atomic (*i.e.* non-compound) proposition or the negation of an atomic proposition.

Literals are the simplest possible formulas, since each literal represents either the assertion or the denial of a nondecomposable statement. The truth table for a literal always has two lines, one for T and one for F.

Example 3.8.1
A, B, $\sim A$, and $\sim D$ are literals.
$\sim\sim A$, $B \vee C$, and $\sim(B \vee C)$ are not literals. Why?

Definition 3.8.6: conjunctive clause
A *conjunctive clause* is a formula that contains only literals and the connective $\wedge$ (if it contains any binary connectives at all), such that no atomic proposition appears more than once.

Example 3.8.2
The following formulas are conjunctive clauses, because none of them contains any connective other than $\wedge$ and $\sim$, because all occurrences of $\sim$ are applied to atomic propositions, and because none of them contains repeated atoms.

$[(\sim A \wedge B) \wedge (C \wedge \sim D)] \wedge E$

$(\sim A \wedge B) \wedge [(E \wedge \sim D) \wedge C]$

$\sim D \wedge [(C \wedge E) \wedge (\sim A \wedge B)]$

A

$\sim C$

Example 3.8.3
The following formulas are not conjunctive clauses.

$A \wedge (\sim B \wedge A)$ and

$(A \wedge C) \wedge (\sim A \wedge B)$

are not conjunctive clauses because each contains multiple occurrences of the atom A.

$(A \wedge C) \leftrightarrow (D \wedge B)$ and

$\sim A \vee C$

are not conjunctive clauses because each contains some binary connective other than $\wedge$.

$(C \wedge \sim A) \wedge \sim(B \wedge \sim D)$ and

$\sim\sim D$

are not conjunctive clauses because each contains an instance of $\sim$ that is not applied to an atomic proposition.

A conjunctive clause is a simple formula, since it asserts that all of its constituent literals are true. If p is a conjunctive clause that contains n literals, then the truth table for p has 2^n lines, all of which are false except one. The true line of the truth table for p is the one in which each of the atoms that appears in p preceded by a $\sim$ sign has the value F and each atom that is not negated in p has the value T.

Practice 3.8.2
Verify this claim, by constructing truth tables for the conjunctive clauses in example 3.8.2.

Conversely, every (non-abbreviated) truth table that has exactly one true line can be represented by some conjunctive clause.

Practice 3.8.3
To convince yourself that this is true, find a conjunctive clause for each of the following truth tables.

A	B	p
F	F	F
F	T	F
T	F	T
T	T	F

A	B	C	q
F	F	F	F
F	F	T	F
F	T	F	T
F	T	T	F
T	F	F	F
T	F	T	F
T	T	F	F
T	T	T	F

Since $\wedge$ is *commutative* (*i.e.* $p \wedge q \equiv q \wedge p$) and *associative* (*i.e.* $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$), any two conjunctive clauses that contain the same set of literals are equivalent, no matter what order the literals appear in, and no matter where the parentheses are. For this reason, we normally write conjunctive clauses without parentheses, and with the literals in some standard order, such as

alphabetical.

Example 3.8.4
The first three conjunctive clauses in example 3.8.2 are equivalent to each other, and each could be written as follows.

$\neg A \wedge B \wedge C \wedge \neg D \wedge E.$

Definition 3.8.7: disjunctive normal form (DNF)
A formula is in ***disjunctive normal form*** (abbreviated *DNF*) if it is a disjunction of (any finite number of) conjunctive clauses. In other words, a DNF formula is formed by connecting a collection of conjunctive clauses together with $\vee$ connectives.

Example 3.8.5
Each of the following formulas is in DNF.

$(\neg A \wedge B) \vee (\neg A \wedge \neg B)$

$(A \wedge B \wedge C) \vee [(A \wedge \neg B) \vee (B \wedge \neg C)]$

$\{A \vee [\neg C \vee (\neg A \wedge B)]\} \vee (\neg B \wedge C \wedge D)$

There are two special cases of DNF formulas that deserve special attention. If the number of conjunctive clauses that are connected together by $\vee$'s is one, then there really is no $\vee$ at all. That is, a conjunctive clause by itself is a special case of a DNF formula. If the number of conjunctive clauses is zero, then there would appear to be no formula at all. We use F to denote this limiting case of a DNF formula, and we understand F to mean a contradictory formula. That is, F is the (only) DNF formula that is always false. (Can you explain why this is so?)

Since $\vee$, like $\wedge$, is commutative and associative, we customarily omit the extra parentheses around the $\vee$'s. The only parentheses that remain, then, are the parentheses around the conjunctive clauses.

Example 3.8.6
The formulas of example 3.8.5 would thus be written as follows.

$(\neg A \wedge B) \vee (\neg A \wedge \neg B)$

$(A \wedge B \wedge C) \vee (A \wedge \neg B) \vee (B \wedge \neg C)$

$(A) \vee (\neg C) \vee (\neg A \wedge B) \vee (\neg B \wedge C \wedge D)$

Disjunctive normal form formulas are "simple", and thus "easy to work with" in many situations, because they have only two "levels" of connectives: the $\wedge$'s in the conjunctive clauses, and the $\vee$'s between them. DNF formulas are also simple in that they contain only two kinds of binary connectives, and that negations are only applied to atomic propositions. We named this class of formulas in anticipation of the following theorem.

Theorem 3.8.1
DNF is a normal form.

Proof
We must show that for every formula $p \in \mathbf{F}$ there is a formula q in DNF that is equivalent to p. What we will do, in fact, is show how to construct q from a truth table for p.

Let $\mathbf{T}$ be the truth table for p. If every row of $\mathbf{T}$ is F, then p is a contradiction, and we can use the DNF formula F for q. Otherwise, we can construct a conjunctive clause to "represent" each row of $\mathbf{T}$ that is true. The disjunction of

these clauses will be q.

Suppose that the j^{th} row of $\mathbf{T}$ is true. Then the conjunctive clause q_j that represents this row contains every atom that labels a column of $\mathbf{T}$. Each atom will be negated in q_j if there is an F in the corresponding column of $\mathbf{T}$ at row j, and it will not be negated if there is a T.

For example, if row j of $\mathbf{T}$ is

A	B	C	D	E	p
T	F	F	T	F	T

then q_j is the following clause.

$A \wedge \neg B \wedge \neg C \wedge D \wedge \neg E$

Note that the truth table for q_j is true only for row j. Therefore, if q is the disjunction of all of the q_j's for the rows of $\mathbf{T}$ that are true, then the truth table for q will be true in exactly the same rows as $\mathbf{T}$ is true. In other words, q has the same truth table as p. Since q is a disjunction of conjunctive clauses, q is a DNF formula equivalent to p, as was to be shown.
■

Example 3.8.7
We will illustrate by constructing a DNF formula equivalent to the following formula.

$\neg(A \leftrightarrow B) \wedge (C \vee \neg A)$

The truth table for this formula looks like this.

A	B	C	$A \leftrightarrow B$	$C \vee \neg A$	$\neg(A \leftrightarrow B) \wedge (C \vee \neg A)$
F	F	F	T	T	F
F	F	T	T	T	F
F	T	F	F	T	T
F	T	T	F	T	T
T	F	F	F	F	F
T	F	T	F	T	T
T	T	F	T	F	F
T	T	T	T	T	F

The DNF formula constructed from this truth table is the following formula.

$(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C)$

A shorter DNF formula that is also equivalent to the original formula is this one.

$(\neg A \wedge B) \vee (A \wedge \neg B \wedge C)$

This example illustrates that there may be several DNF formulas that are equivalent to a given formula, and that the construction we have given may not produce the shortest such DNF formula. The problem of finding a shortest DNF formula equivalent to a given formula is an interesting and difficult combinatorial problem.

Practice 3.8.4
Convert the following formulas to DNF.

$(A \longrightarrow B) \longrightarrow (B \longrightarrow A)$

$(A \wedge B) \vee (B \wedge C)$

$\neg(A \vee B \vee C \vee D)$

$$(A \longrightarrow B) \longleftrightarrow [\sim(\sim A \vee \sim B) \longrightarrow (B \vee \sim C)]$$

The dual concept to disjunctive normal form is the concept of conjunctive normal form. The development that follows is completely analogous to the development of DNF above, except that the roles of $\wedge$ and $\vee$ are interchanged and T and F are interchanged.

Definition 3.8.8: disjunctive clause

A ***disjunctive clause*** is a formula that contains only literals and the connective $\vee$ (if it contains any binary connectives at all), such that no atomic proposition appears more than once.

Example 3.8.8

A

$\sim B$

$A \vee \sim C \vee E$ and

$\sim B \vee C$

are all disjunctive clauses.

A disjunctive clause is simple, because it merely asserts that at least one of a list of undecomposable propositions (or their negations) is true. Another way to look at it is that a disjunctive clause is false if and only if each of its literals is false.

Definition 3.8.9: conjunctive normal form (CNF)

A formula is in ***conjunctive normal form*** (abbreviated *CNF*) if it is a conjunction of (any finite number of) disjunctive clauses. In other words, a CNF formula is formed by connecting a collection of disjunctive clauses together with $\wedge$ connectives.

Similar to DNF, a disjunctive clause by itself is a special case of a CNF formula. The empty CNF formula is represented by T, which we use to stand for (and be equivalent to) any tautology. Also similar to DNF, we have the following theorem.

Theorem 3.8.2

CNF is a normal form.

Proof

The proof is completely analogous to the proof that DNF is a normal form. We just have to exchange the roles of $\wedge$ with $\vee$ and T with F throughout the proof.

Given an arbitrary formula p, we construct a CNF formula q that is equivalent to p by taking the conjunction of the set of disjunctive clauses that "represent" the rows of the truth table for p that are false. (If all rows of the truth table for p are true, then p is a tautology, and we set q to be T.)

We construct the disjunctive clause that "represents" a particular row of the truth table by making sure the clause is false only for that row. For example, to represent the truth table row

A	B	C	D	E	p
F	T	F	T	F	F

we would construct the following disjunctive clause.

$$A \vee \sim B \vee C \vee \sim D \vee E$$

If q is constructed from clauses that are generated in this way, then q will be a CNF formula equivalent to p.

■

Theorems 3.8.1 and 3.8.2 could also be proved by showing how to construct the DNF or CNF formula q by successively transforming p using DeMorgan's laws and the distributive laws. The formula p may have to be first transformed by replacing subformulas containing other connectives into equivalent subformulas containing only $\wedge$, $\vee$, and $\sim$. Although this method of constructing q can be complicated, it sometimes leads to shorter DNF or CNF formulas.

Observe that if a formula q is in DNF, it is very easy to find an assignment of truth values (T or F) to the propositions in q that makes q true. We simply have to make any one of the conjunctive clauses in q true. If q is in CNF, how can you find an assignment of truth values that makes q false?

Example 3.8.9

The CNF formula equivalent to the formula of example 3.8.7 is this formula.

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

A shorter equivalent CNF formula is this one.

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

Practice 3.8.5

Find a different equivalent CNF formula that is as short as the second formula in the above example.

Practice 3.8.6

Convert each of the formulas in practice 3.8.4 to CNF.

Disjunctive normal form and conjunctive normal form find application in a variety of areas. As we will see later, they play a role in the design of the switching circuits from which computers are constructed. They are also essential in the design of *mechanical theorem provers*, programs that try to distinguish tautologies from non-tautologies more efficiently than by trying to evaluate the formula for all of the 2^n possible rows of the truth table. Mechanical theorem proving plays an essential role in several areas of computing, including program verification (proving that computer programs are "correct"), program generation (generating computer programs from high level descriptions), question answering systems, robotics, and natural language understanding. The most well-known theorem proving method, called ***resolution***, works by manipulating formulas in CNF. The importance of circuit design and resolution theorem proving to computing science gives two strong reasons to become familiar with DNF and CNF.

Section 3.9. Circuit Diagrams

We have represented the semantics of propositions by truth tables -- talking about when formulas are true or false based on their components. This is not the only possible "interpretation" of propositional logic. An equally good one -- one which is of course equivalent to a truth table analysis -- is in terms of electrical circuits. This topic is covered in Chapter 10.1.

Section 3.10. Syntax, Semantics and the Deduction Theorem

A logical implication is, intuitively, a relation between the premises of an argument and the conclusion of that argument. To say that the logical implication holds, one is saying that the argument is valid. That is, the relation between premises and conclusion is one of "must follow" or "necessarily follows." It is a case of where it is necessary that *if* the premises are true then the conclusion will also be true. Or, to put it another way, it would be contradictory to accept the premises and reject the conclusion. Again intuitively, if you grant the premises then you must also grant the conclusion. The relation of logical implication is different from the relation of $\longrightarrow$. The $(p \longrightarrow q)$ relation merely says that if p is (actually) true, then q will be true also. The relation of logical implication says that *every* time (that is, in each row of the truth table) the premises are all true, so is the conclusion. Clearly there is some correspondence between these two notions, but they are not exactly the same. This correspondence is explained by the deduction theorem, which we will explain in a few paragraphs from now.

In the previous paragraph we have slid back and forth between two types of ideas: on the one hand we spoke of "being true", or "being necessarily true" or "being contradictory", and generally of matters relevant to the truth or falsity of statements; while on the other hand we spoke of "must follow" and "necessarily follows", and generally about matters having to do with (as we shall shortly see) the logical forms of the sentences involved. When presented in terms of "must follow" (such as when one says that in a good argument a conclusion "must follow" from the premises), this notion of logical implication amounts to asserting that the sentences involved exhibit one of a certain class of patterns, or obey certain rules. Such a characterization does not *directly* have anything to do with the truth or falsity of the sentences involved in the implication. This characterization of logical implication as a matter of exhibiting certain patterns or obeying certain rules is called a ***syntactic characterization*** (or: the ***syntax***) of the logic. The characterization in terms of truth values, tautologies, etc., is called a ***semantic characterization*** (or: the ***semantics***) of the logic.

In the semantic characterization of the logic, we are interested in which formulas are tautologies -- always true no matter what. Or if we are talking about arguments, we are interested in whether it is the case that if the premises are all true, so is the conclusion. In the syntactic characterization we are interested in the notion of a *proof* of a formula (or of the proof of a conclusion from a set of premises), without addressing truth values at all. Instead, it is a way of displaying that the formula can be generated by following a certain set of rules. It is something which needs to be established that every formula which can be proved is a tautology, and that every tautology is a formula which can be proved; and that every argument which is syntactically provable is also semantically valid, and the reverse. We shall investigate this correspondence between the semantics and the syntax in Chapter 9. (These two notions are called the ***soundness*** and the ***completeness*** of the logic, respectively. So ***soundness*** means that if it is syntactically provable, then it is semantically valid; ***completeness*** means that if it is semantically valid, then it is provable. For now we shall just assume that there is such a true correspondence. It is proved in Chapter 9.)

Here's a little notation: we shall use '$\Gamma \models p$' to stand for "the logical implication between the set of premises Γ and the conclusion p is valid". That is, if the premises in Γ are true, so is the conclusion p. As a special case, when Γ is empty, we write $\models p$ to indicate that p is a tautology. We might note that if $\Gamma \models p$ then the conditional formed by the conjunction of each member of Γ as antecedent and p as consequent will be a tautology. For example, if q, r and s are all the sentences in Γ, then $\Gamma \models p$ if and only if $\models((q \wedge r \wedge s) \longrightarrow p)$. (You should check this claim out by constructing the relevant truth tables.) This is (the semantic version of) ***the deduction theorem***, stating

a relationship to hold between arguments (with premises) and the corresponding conditionals' being tautologies. We use the notation '$\Gamma \vdash p$' to mean that p follows (syntactically) from the set of formulas in Γ as premises. Again as a special case where Γ is empty, we use $\vdash p$ to indicate that p can be proved from zero premises.

As mentioned before, there is a correspondence between the syntax and the semantics of propositional logic. Just as there is a semantic deduction theorem, so too is there a syntactic deduction theorem saying: $\Gamma \vdash p$ if and only if the corresponding conditional can be proved (without premises). Again, if q, r and s are all the formulas in Γ, then $\Gamma \vdash p$ if and only if $\vdash[(q \wedge r \wedge s) \longrightarrow p]$.

So we see that both syntactically and semantically there is a way of converting "argument forms" -- i.e., logical implications -- into individual sentences, and vice versa. Sometimes it is handy to do this, as we shall see in the syntactic case; but other times we can adapt our existing methods to handle the more general case of having premises and conclusions. For example, we *could* use the deduction theorem with our tree method of evaluating sentences. All we would have to do is convert the argument with premises into the corresponding conditional, and then apply the tree method of negating this conditional and branching. But it would be just as easy, or easier, to follow the method outlined in the section on arguments: make the root of the tree contain all the premises and the negation of the conclusion.

Section 3.11. An Explicit Formal System of Propositional Logic

A syntactic presentation of a system of logic is generally made by stating

(1) a set of axioms

(2) a set of rules of inference

(3) a definition of a proof, that is, a definition of "from premises p_1, p_2, $\cdots$ p_m we can prove q".

Generally speaking, an axiom is a formula (or formula-type) which is simply presented without justification, and a rule of inference is a way of converting (one or more) already presented formulas into another (and thus allowing the rules of inference to operate on it). Using this method, one usually defines "a proof of q from $p_1, p_2, \cdots p_m$" (remember we abbreviated this as $\Gamma \vdash q$, where $p_1, p_2, \ldots p_m$ are the members of Γ) like this: the proof is a finite sequence of lines of formulas, each one of which either (a) is a member of Γ, or (b) is an axiom, or (c) is the result of applying a rule of inference to previous members of the sequence; and (d) the formula q occurs as the last line of the sequence. Such systems of logic are called ***axiomatic systems***.

Now, as it turns out, it is possible for there to be systems of logic with no axioms, at the cost of having a larger set of rules of inference and a somewhat more complicated definition of "proof". Such systems are called ***natural deduction systems***, and are what we shall present here.

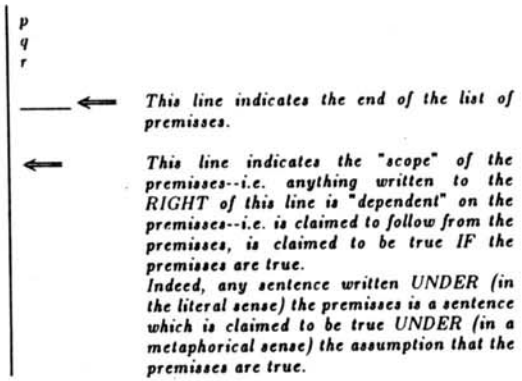
Certain rules of inference are quite reasonable and natural, for example, if you have $(p \wedge q)$ at your disposal then you should be allowed to infer p (and also infer q). The reason that this rule is natural is because of the meaning, or point, of '$\wedge$', which is to be understood as 'and'. Similarly, if we are presented with $\sim\sim p$ we should be allowed to infer p ("from a double negative, infer the positive"). Another obvious rule of inference is that if you have been given two things, p and $(p \longrightarrow q)$ then you should be able to infer q. A different kind of rule of inference, but one which is still intuitively valid, goes like this: "I want to prove that p is true. Just imagine what

would happen if it weren't. Why, then q would be true, and hence r would be false and so s would have to be true. But this is impossible because s is contradictory. So our assumption that p is not true has to be wrong; therefore p is true." This method of argumentation is extremely popular and useful; it is often called an *indirect proof* (of p) or a *reductio ad absurdum* (of not-p) or a *proof by contradiction* (of p). One version of this rule was incorporated in our truth trees, when we negated the conclusion and discovered that all paths closed. (That was our contradiction). We concluded that the assumption of the negation had to be wrong.

Let us investigate the structure of this last argument a little more. We already knew some facts (e.g., that *s* was false), and we wished to prove from these known facts that *p* had to be true. What we did here was to *temporarily* make the assumption that *p* was false (i.e., that $\sim p$ was true), and then we looked at what this assumption led to. What we discovered was that it led to a contradiction. Since it leads to a contradiction, the assumption $\sim p$ cannot be right. So we should (and did) infer from this chain of reasoning (from this *subargument*), that *p* must be right. But also we have to now give up our assumption -- after all it was just an assumption for a subargument, to see where it would lead. The justification for *p* is the entire subargument, but once we have inferred *p* we need to "erase" or "eliminate" or at least never be able to use our assumption $\sim p$ (and also of course not use any of the reasoning which started with $\sim p$).

Clearly the use of assumptions is a familiar, legitimate and important part of ordinary argumentation, but just as clearly, to *assume* something (for the purpose of argument) is quite different from *proving* it. So an important feature of any natural deduction system ought to be a clear and simple device for distinguishing sentences which are assumed from sentences which are derived and for making it clear just what assumptions (if any) a sentence is "dependent" on. There are many ways in which this can be done; the way that we will choose is the simplest of those of which we are aware. Before we turn to that device, we may begin with some simpler considerations.

Suppose we have as premisses of an argument p, q, and r. We begin by listing them in the following manner:

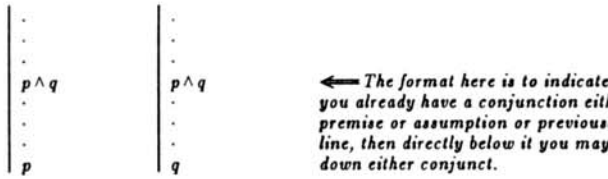


Before proceeding to discuss further the format for handling additional assumptions

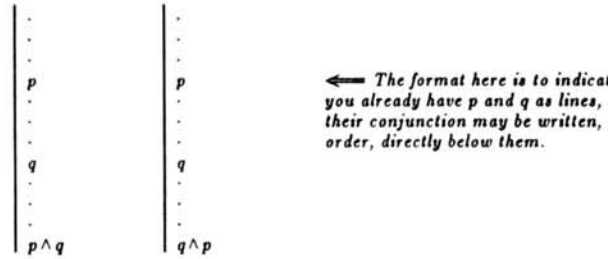
made in the course of an argument (or *derivation*, as we shall call our formal arguments), we consider a few simple rules and a few simple derivations.

We introduce eleven rules; here are a few of the simplest ones:

CONJUNCTION ELIMINATION ($\wedge E$)



CONJUNCTION INTRODUCTION ($\wedge I$)



Each of these two rules has two forms. In the rule ($\wedge E$) we can "simplify" or "eliminate" from either side of a conjunction. In the ($\wedge I$) rule, we can form our conjunction in either direction, so long as we have both conjuncts available. When writing a proof, you not only apply these rules of inference, but also you include another column (which is called the *annotation*) in which you state what previous lines were used and which rule of inference was applied. It should also be noted that these rules can be used on a line of a proof which was an assumption (the formula above a horizontal line). In presenting a proof we give line numbers off to the left so that we can refer to preceding lines.

Example: We derive $P \wedge Q$ from $Q \wedge P$:

1.	$Q \wedge P$	

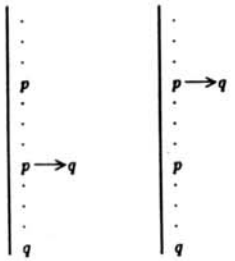
2.	Q	1, $\wedge E$
3.	P	1, $\wedge E$
4.	$P \wedge Q$	2,3 $\wedge I$

Example: We derive $P \wedge (Q \wedge R)$ from $(R \wedge Q) \wedge R$:

1.	$(P \wedge Q) \wedge R$	

2.	$P \wedge Q$	1, $\wedge E$
3.	P	2, $\wedge E$
4.	Q	2, $\wedge E$
5.	R	1, $\wedge E$
6.	$Q \wedge R$	4,5 $\wedge I$
7.	$P \wedge (Q \wedge R)$	3,6 $\wedge I$

IMPLICATION ELIMINATION ($\rightarrow E$) (Traditionally known as Modus Ponens)



Example: We derive R *from* P, $P \rightarrow Q$, $Q \rightarrow R$:

1.	P	
2.	$P \rightarrow Q$	
3.	$Q \rightarrow R$	

4.	Q	1,2 $\rightarrow E$
5.	R	3,4 $\rightarrow E$

Example: We derive R from $P \vee S$, $(P \vee S) \rightarrow Q$, $Q \rightarrow R$:

1.	$P \vee S$	
2.	$(P \vee S) \rightarrow Q$	
3.	$Q \rightarrow R$	

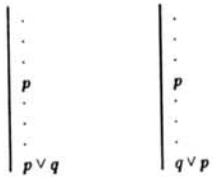
4.	Q	1,2 $\rightarrow E$
5.	R	3,4 $\rightarrow E$

Example: We derive R from P, Q, $(P \wedge Q) \rightarrow (R \wedge S)$:

1.	P	
2.	Q	
3.	$(P \wedge Q) \rightarrow (R \wedge S)$	

4.	$P \wedge Q$	1,2 $\wedge I$
5.	$R \wedge S$	3,4 $\rightarrow E$
6.	R	5, $\wedge E$

DISJUNCTION INTRODUCTION ($\vee I$)



Example: We derive R from P and $(P \vee Q) \rightarrow R$:

1.	P	
2.	$(P \vee Q) \rightarrow R$	

3.	$P \vee Q$	1, $\vee I$
4.	R	2,3 $\rightarrow E$

NEGATION ELIMINATION ($\sim E$) (often called Double Negation)

Example: We derive $P \wedge Q$ *from* $P \wedge \sim\sim Q$:

1.	$P \wedge \sim\sim Q$	
2.	P	1, $\wedge E$
3.	$\sim\sim Q$	1, $\wedge E$
4.	Q	3, $\sim E$
5.	$P \wedge Q$	2,4 $\wedge I$

Example: We derive S from Q, $P \wedge T$, $((P \wedge Q) \vee R) \rightarrow \sim\sim S$:

1.	Q	
2.	$P \wedge T$	
3.	$((P \wedge Q) \vee R) \rightarrow \sim\sim S$	
4.	P	2, $\wedge E$
5.	$P \wedge Q$	1,4 $\wedge I$
6.	$(P \wedge Q) \vee R$	5, $\vee I$
7.	$\sim\sim S$	3,6 $\rightarrow E$
8.	S	7, $\sim E$

It is important to identify the main connective of a sentence when using inference rules-- the inference rules never apply to internal parts of a sentence. For example, you can never derive a conjunction by disjunction introduction, or derive something from a disjunction by implication elimination, even if there are conjunctions or disjunctions embedded within the formulas.

Here are some examples of the *misuse* of the inference rules. In the first one, we incorrectly did a $\vee I$ on the antecedent, rather than to the whole formula. In the second we incorrectly did $\rightarrow E$ on a subpart of the formula on line 2 (to correctly use $\rightarrow E$ on line 2 we would need the formula $P \rightarrow Q$, and then we would infer R rather than $(Q \rightarrow R)$. In the third example, we incorrectly applied $\wedge E$ to an embedded occurrence of $\wedge$. And in the last example we applied $\wedge E$ to the antecedent of line 1, whereas the main connective is $\rightarrow$.

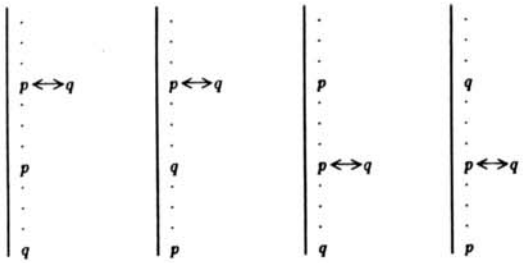
1.	$P \rightarrow Q$	
2.	$(P \vee R) \rightarrow Q$	*INCORRECT* $\vee I$ *on 1*

1.	$((P \rightarrow Q) \rightarrow R) \wedge P$	
2.	$(P \rightarrow Q) \rightarrow R$	1, $\wedge E$
3.	P	1, $\wedge E$
4.	$Q \rightarrow R$	*INCORRECT* $\rightarrow E$ *on 2,3*

1.	$(P \wedge Q) \wedge R$	
2.	P	*INCORRECT* $\wedge E$ *on 1*

1.	$(P \wedge Q) \rightarrow R$	
2.	$Q \rightarrow R$	*INCORRECT* $\wedge E$ *on 1*

EQUIVALENCE ELIMINATION ($\leftrightarrow E$)

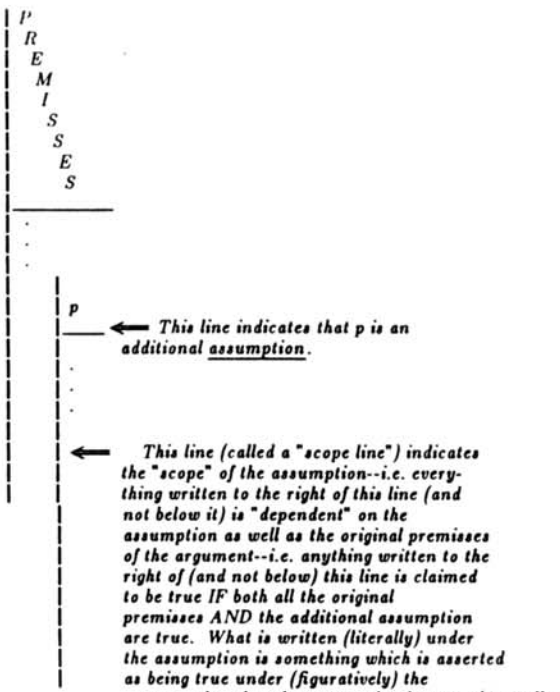


Example: we derive Q from $P \leftrightarrow (Q \leftrightarrow R)$ *and* $P \wedge R$:

1.	$P \leftrightarrow (Q \leftrightarrow R)$	
2.	$P \wedge R$	
3.	P	2, $\wedge E$
4.	$Q \leftrightarrow R$	1,3 $\leftrightarrow E$
5.	R	2, $\wedge E$
6.	Q	4,5 $\leftrightarrow E$

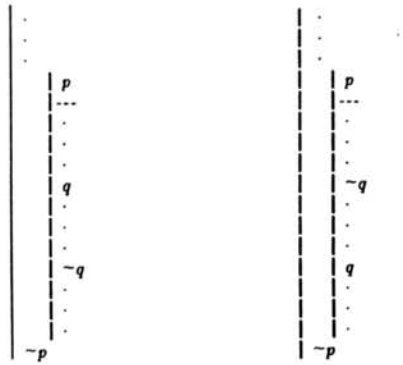
We now turn to the problem of formulating a rule of *Negation Introduction* -- there are a number of ways this could be done, but our choice will be a form of Reductio Ad Absurdum argument. In such a case we prove $\sim p$ by *assuming* p and deriving a contradiction. On the basis of the existence of the subargument or *subderivation* we conclude that $\sim p$, simultaneously giving up our assumption of p.

We noted earlier that it is important to distinguish assumptions from derived sentences etc. To achieve this we further employ an earlier device--we indicate that we are *assuming* p by the following:



Using this new device, we may represent negation introduction as follows:

NEGATION INTRODUCTION (~I)



Note: $\sim p$ is *not* written under the assumption p as it is not dependent on it; instead it is "popped out" one level.

Note: As a special case either q or $\sim q$ may be p itself.

In arguing under an assumption we do, of course, refer back to things we have already proved. For example, we derive $\sim P$ from $P \rightarrow Q$ and $P \rightarrow \sim Q$:

1.	$P \rightarrow Q$	
2.	$P \rightarrow \sim Q$	

3.	P	

4.	$P \rightarrow Q$	1, 3
5.	$P \rightarrow \sim Q$	2, 3
6.	Q	3,4 $\rightarrow$E
7.	$\sim Q$	3,5 $\rightarrow$E
8.	$\sim P$	3-7 ~I

Note: In the last line we refer to the WHOLE SUBDERIVATION 3-7, RATHER THAN TO SPECIFIC LINES, because it is the existence of the subderivation that justifies the conclusion $\sim P$.

But how do we justify lines 4 and 5? Intuitively, they *are* justified, because we are still arguing under our original premisses, and certainly anything we were entitled to assert on the basis of those premisses, we are still entitled to assert even though we now have an additional assumption.

We introduce the rule of Reiteration (R) to allow us to repeat lines already derived from the premisses (or the premisses themselves) or lines already derived from the premisses and additional assumptions *provided that* we are still arguing under the scope of those assumptions. (We could in fact eliminate this rule, but only at the cost of making certain rules that we will introduce in the next chapter very messy to state.)

Intuitively, the rule is straightforward--any premise or assumption, or anything derived from the premisses plus previous assumptions may be asserted at any point, *provided that* we are still arguing under the scope of those previous assumptions.

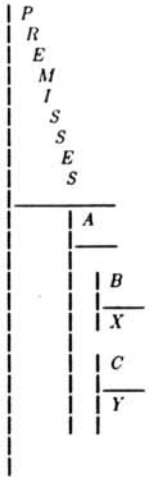
Although intuitively clear, the rule of reiteration is a little difficult to state rigorously.

REITERATION (R)

> A line X of a derivation may be written again as line Y provided that every scope line to the left of X continues unbroken to the left of Y and provided that X occurs above Y in the derivation.

Don't be put off by this definition; the idea behind it is simple: You can reiterate any line previously occurring in a derivation provided that all the assumptions "under" which it occurred are still "operative". For those of you who have programmed in block structured languages, this should be familiar. Reiteration (until a new wrinkle that comes up in the next chapter) is merely the usual statement of when an identifier is valid: it is valid in its own block and in any block that it contains. The scope lines in this system of propositional logic can therefore be viewed like the blocks in a statically-scoped programming language.

Reiteration is *never* done from below or from the right. It may occur only from above or from above and to the left, and *not always* then as the following example illustrates:



Here line X may not be reiterated at line Y, because there is a scope line to the left of X which does not continue unbroken to the left of Y. And intuitively speaking, it would wrong to do so since X may have depended on the assumption B, whereas line Y is in a position that is not dependent on B.

Now that we have our reiteration rule we can complete the annotation of the derivation on the previous page:

1.	$P \rightarrow Q$	
2.	$P \rightarrow \sim Q$	
3.	P	
4.	$P \rightarrow Q$	1, R
5.	$P \rightarrow \sim Q$	2, R
6.	Q	3,4 $\rightarrow E$
7.	$\sim Q$	3,5 $\rightarrow E$
8.	$\sim P$	3,7 $\sim I$

Example: We derive $\sim P$ from $P \rightarrow Q$ and $\sim Q$:

1.	$P \rightarrow Q$	
2.	$\sim Q$	
3.	P	
4.	$P \rightarrow Q$	1, R
5.	Q	3,4 $\rightarrow E$
6.	$\sim Q$	2, R
7.	$\sim P$	3-6, $\sim I$

Example: We give two different derivations of $\sim Q$ from $P \rightarrow (Q \rightarrow R)$ and $P \wedge \sim R$:

1.	$P \rightarrow (Q \rightarrow R)$		1.	$P \rightarrow (Q \rightarrow R)$	
2.	$P \wedge \sim R$		2.	$P \wedge \sim R$	
	---			---	
3.	P	2, $\wedge E$	3.	$\quad Q$	
4.	$\sim R$	2, $\sim E$	4.	$\quad$ ---	2, R
5.	$Q \rightarrow R$	1,3 $\rightarrow E$	5.	$\quad P$	4, $\wedge E$
6.	$\quad Q$		6.	$\quad P \rightarrow (Q \rightarrow R)$	1, R
	$\quad$ ---			$\quad$	
7.	$\quad Q \rightarrow R$	5, R	7.	$\quad Q \rightarrow R$	5,6 $\rightarrow E$
8.	$\quad R$	6,7 $\rightarrow E$	8.	$\quad R$	3,7 $\rightarrow E$
9.	$\quad \sim R$	4, R	9.	$\quad \sim R$	4, $\wedge E$
10.	$\sim Q$	6-9 $\sim I$	10.	$\sim Q$	3-9 $\sim I$

Negation Elimination and Negation Introduction can be used together to create another sort of Reductio Ad Absurdum argument: as the following two examples illustrate, we can sometimes prove a formula p by assuming $\sim p$, deriving a contradiction, concluding $\sim\sim p$ by NI, and then p by NE.

Example: We derive P from $\sim P \rightarrow Q$ and $\sim Q$:

1.	$\sim P \rightarrow Q$	
2.	$\sim Q$	

3.	$\quad \sim P$	
	$\quad$ ---	
4.	$\quad \sim P \rightarrow Q$	1, R
5.	$\quad Q$	3,4 $\rightarrow E$
6.	$\quad \sim Q$	2, R
7.	$\sim\sim P$	3-6, $\sim I$
8.	P	7, $\sim E$

Example: We derive $P \vee Q$ from $\sim P \rightarrow Q$:

1.	$\sim P \rightarrow Q$	

2.	$\quad \sim (P \vee Q)$	
	$\quad$ ---	
3.	$\quad P$	
	$\quad$ ---	
4.	$\quad P \vee Q$	3, $\vee I$
5.	$\quad \sim (P \vee Q)$	2, R
6.	$\quad \sim P$	3-5, $\sim I$
7.	$\quad \sim P \rightarrow Q$	1, R
8.	$\quad Q$	6,7 $\rightarrow E$
9.	$\quad P \vee Q$	8, $\vee I$
10.	$\sim\sim (P \vee Q)$	2-9, $\sim I$
11.	$P \vee Q$	10, $\sim E$

We are now in a position to provide derivations for more interesting arguments. Consider the following argument in English.

> God, if he exists, is omnipotent, and omniscient as well. Moreover, he is benevolent, provided that he exists. If God can prevent evil, then if he knows that evil exists, he is not benevolent if he does not prevent it. If he is omnipotent, he can prevent evil. And if he is omniscient, he knows that evil exists if it does exist. Evil does not exist if God prevents it. However, evil does exist. Therefore, God does not exist.

Using the following scheme of abbreviation

G: God exists
P: God is omnipotent
S: God is omniscient
B: God is benevolent
C: God can prevent evil
K: God knows that evil exists
V: God does prevent evil
E: Evil exists

We can translate the premises of the argument as

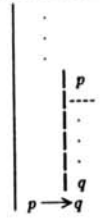
$G \rightarrow P \wedge S$
$G \rightarrow B$
$C \rightarrow (K \rightarrow (\sim V \rightarrow \sim B))$
$P \rightarrow C$
$S \rightarrow (E \rightarrow K)$
$V \rightarrow \sim E$
E

And the conclusion as $\sim G$. (You should try to do this to see if you get the same translations). A derivation for this argument is:

1.	$G \rightarrow P \wedge S$	
2.	$G \rightarrow B$	
3.	$C \rightarrow (K \rightarrow (\sim V \rightarrow \sim B))$	
4.	$P \rightarrow C$	
5.	$S \rightarrow (E \rightarrow K)$	
6.	$V \rightarrow \sim E$	
7.	E	
8.	| G	
9.	| $G \rightarrow P \wedge S$	1, R
10.	| $P \wedge S$	8,9 $\rightarrow E$
11.	| P	10, $\wedge E$
12.	| S	10, $\wedge E$
13.	| $G \rightarrow B$	2, R
14.	| B	8,13 $\rightarrow E$
15.	| $P \rightarrow C$	4, R
16.	| C	11,15 $\rightarrow E$
17.	| $C \rightarrow (K \rightarrow (\sim V \rightarrow \sim B))$	3, R
18.	| $K \rightarrow (\sim V \rightarrow \sim B)$	16, 17 $\rightarrow E$
19.	| $S \rightarrow (E \rightarrow K)$	5, R
20.	| $E \rightarrow K$	12, 19 $\rightarrow E$
21.	| E	7, R
22.	| K	20, 21 $\rightarrow E$
23.	| $\sim V \rightarrow \sim B$	18, 22 $\rightarrow E$
24.	| | V	
25.	| | $V \rightarrow \sim E$	6, R
26.	| | $\sim E$	24, 25 $\rightarrow E$
27.	| | E	7, R
28.	| $\sim V$	24-27 $\sim I$
29.	| $\sim B$	23, 28 $\rightarrow E$
30.	$\sim G$	8-29 $\sim I$

The next rule we introduce is implication introduction. Often we want to derive conclusion which is a conditional statement. For example, we might want to derive $P \rightarrow R$ from $P \rightarrow Q$ and $Q \rightarrow R$; or $(P \wedge Q) \rightarrow R$ from $P \rightarrow (Q \rightarrow R)$. The normal technique employed in proving a conditional (frequently called 'conditional proof') is to assume the antecedent of the condtional that we wish to prove and to derive the consequent from it, concluding, on the basis of this subderivation, that the conditional holds. This suggests the following rule:

IMPLICATION INTRODUCTION ($\rightarrow I$)



Example: We derive $P \rightarrow R$ from $P \rightarrow Q$ and $Q \rightarrow R$:

1.	$P \rightarrow Q$	
2.	$Q \rightarrow R$	
3.	| P	
4.	| $P \rightarrow Q$	1, R
5.	| Q	3,4 $\rightarrow E$
6.	| $Q \rightarrow R$	2, R
7.	| R	5,6 $\rightarrow E$
8.	$P \rightarrow R$	3-7 $\rightarrow I$

(Again we cite an entire subderivation, as it is its existence which justifies our conclusion.)

Example: We derive $Q \rightarrow P$ from P:

1.	P	
2.	| Q	
3.	| P	1, R
4.	$Q \rightarrow P$	2-3 $\rightarrow I$

Example: We derive $P \rightarrow Q$ from $\sim P$:

1.	$\sim P$	
2.	| P	
3.	| | $\sim Q$	
4.	| | P	2, R
5.	| | $\sim P$	1, R
6.	| $\sim\sim Q$	3-5 $\sim I$
7.	| Q	6, $\sim E$
8.	$P \rightarrow Q$	2-7 $\rightarrow I$

Example: We derive $(P \wedge Q) \rightarrow R$ from $P \rightarrow (Q \rightarrow R)$:

1.	$P \rightarrow (Q \rightarrow R)$	
2.	$\quad P \wedge Q$	
3.	$\quad P$	2, $\wedge E$
4.	$\quad P \rightarrow (Q \rightarrow R)$	1, R
5.	$\quad Q \rightarrow R$	3,4 $\rightarrow E$
6.	$\quad Q$	2, $\wedge E$
7.	$\quad R$	5,6 $\rightarrow E$
8.	$(P \wedge Q) \rightarrow R$	2-7 $\rightarrow I$

Example: We derive $P \rightarrow (Q \rightarrow R)$ from $(P \wedge Q) \rightarrow R$:

1.	$(P \wedge Q) \rightarrow R$	
2.	$\quad P$	
3.	$\quad\quad Q$	
4.	$\quad\quad P$	2, R
5.	$\quad\quad P \wedge Q$	3,4 $\wedge I$
6.	$\quad\quad (P \wedge Q) \rightarrow R$	1, R
7.	$\quad\quad R$	5,6 $\rightarrow E$
8.	$\quad Q \rightarrow R$	3-7 $\rightarrow I$
9.	$P \rightarrow (Q \rightarrow R)$	2-8 $\rightarrow I$

Sometimes we can derive a sentence from no premises at all. This is possible if and only if the derived sentence is a *theorem* (of propositional logic). As we indicated in the section above on "Syntax, Semantics and the Deduction Theorem", it can be proved that $\vdash p$ if and only if $\models p$. That is, a formula can be proved from no premises ($\vdash p$) just in case it is a tautology ($\models p$). In such a case, since we have no premisses, we merely use a vertical line without a horizontal line.

Example: We derive $P \rightarrow P$:

1.	$\quad P$	
2.	$\quad P$	1, R
3.	$P \rightarrow P$	1-2 $\rightarrow I$

Example: We derive $P \rightarrow (Q \rightarrow P)$:

1.	$\quad P$	
2.	$\quad\quad Q$	
3.	$\quad\quad P$	1, R
4.	$\quad Q \rightarrow P$	2-3 $\rightarrow I$
5.	$P \rightarrow (Q \rightarrow P)$	1-4 $\rightarrow I$

Example: We derive $\neg(P \wedge \neg P)$:

1.	$\quad P \wedge \neg P$	
2.	$\quad P$	1, $\wedge E$
3.	$\quad \neg P$	1, $\wedge E$
4.	$\neg(P \wedge \neg P)$	1-3 $\neg I$

Example: We derive $(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$:

1.	$\quad P \rightarrow (Q \rightarrow R)$	
2.	$\quad\quad P \rightarrow Q$	
3.	$\quad\quad\quad P$	
4.	$\quad\quad\quad P \rightarrow Q$	2, R
5.	$\quad\quad\quad Q$	3,4 $\rightarrow E$
6.	$\quad\quad\quad P \rightarrow (Q \rightarrow R)$	1, R
7.	$\quad\quad\quad Q \rightarrow R$	3,6 $\rightarrow E$
8.	$\quad\quad\quad R$	5,7 $\rightarrow E$
9.	$\quad\quad P \rightarrow R$	3-8 $\rightarrow I$
10.	$\quad (P \rightarrow Q) \rightarrow (P \rightarrow R)$	2-9 $\rightarrow I$
11.	$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$	1-10 $\rightarrow I$

The following rule should be intuitively plausible.

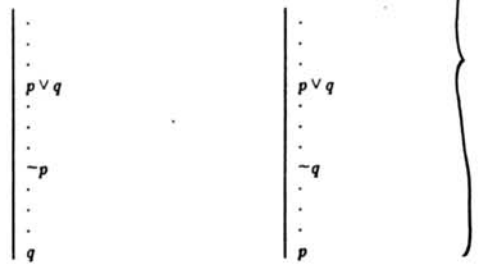
EQUIVALENCE INTRODUCTION ($\leftrightarrow I$)

Normally, when trying to prove an equivalence, you do not have at your disposal the conditionals each way. Therefore, it is normal to use $\leftrightarrow I$ in conjunction with $\rightarrow I$. One recognizes that it is desired to prove the equivalence, and so sets out first to prove each conditional.

Example: We derive $P \leftrightarrow \sim\sim P$

1.	P	
2.	$\quad \sim P$	
3.	$\quad P$	1, R
4.	$\sim\sim P$	2-3 $\sim I$
5.	$(P \rightarrow \sim\sim P)$	1-4 $\rightarrow I$
6.	$\sim\sim P$	
7.	P	7 $\sim E$
8.	$\sim\sim P \rightarrow P$	6-7 $\rightarrow I$
9.	$P \leftrightarrow \sim\sim P$	5,8 $\leftrightarrow I$

Our final rule of inference is disjunction elimination. There are various ways to state such a rule, all of which are equivalent in the presence of our other rules. Later we shall present some of these others as "derived rules". We choose what we think is the most simple version of disjunction elimination.



In this course, this rule is not allowed, unless you prove it using ∨-Elim.

The intuitive justification for such a rule is this: the $p \vee q$ line says "at least one of p and q are true", and the negation line says that it isn't p that's true (or, it isn't q that's true). So it must be the other one which is true.

We now have all the rules required in order to be able to derive the conclusion of any truth functionally valid argument from its premisses. (We haven't proved this but you can take our word for it.)

In order to "abbreviate" proofs, we could show that from these rules other "derived" rules are also acceptable. If we were to do a lot of proofs, it would be worth our while to prove ten or fifteen derived rules. We shall content ourselves with mentioning some of the most common other rules. One familiar argument form employs 'sub-arguments'. This is frequently called "Argument by Cases".

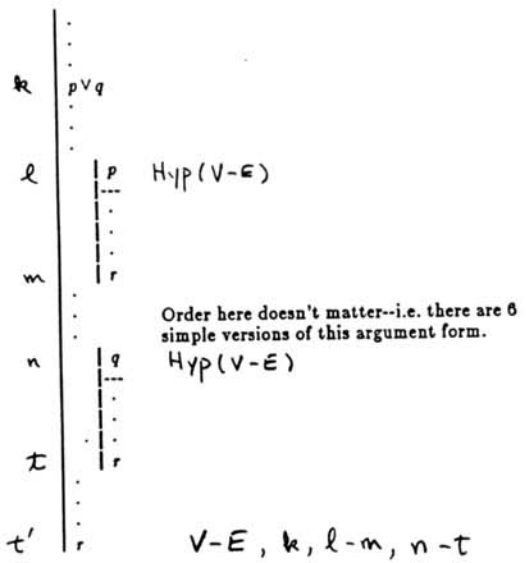
Example: Either John's mother or his girlfriend will be at the recital. If his girlfriend is at the recital, and John makes noise, he will be unable to hear the music, and so will be displeased. John doesn't want to displease his girlfriend. If his mother is at the recital and John makes noise, whe will be mortified and punish him. Therefore, if John is smart he'll keep quiet at the recital.

This argument depends on many implicit premisses, and so is not directly amenable to a truth functional analysis, but one would naturally argue to the conclusion by showing that it follows in either case--i.e. whether his girlfriend or his mother attends the recital. Note that the order in which the two cases are presented does not matter, and neither does it matter whether the "either...or..." is stated before or after or between the two subproofs.

Argument by cases is particularly common in mathematical contexts.

Our comments are suggestive of the following rule:

DISJUNCTION ELIMINATION ($\vee E$)



Order here doesn't matter--i.e. there are 6 simple versions of this argument form.

Example: We derive Q from $P \rightarrow Q$, $R \rightarrow Q$ and $P \vee R$:

1.	$P \rightarrow Q$		
2.	$R \rightarrow Q$		
3.	$P \vee R$		

4.	| P		
	|---		
5.	| $P \rightarrow Q$	1, R	
6.	| Q	4,5 $\rightarrow E$	
7.	| R		
	|---		
8.	| $R \rightarrow Q$	2, R	
9.	| Q	7,8 $\rightarrow E$	
10.	Q	3, 4-6, 7-9 SC	⟸ here we cite the disjunction and the two subderivations.

Example: We derive $Q \vee P$ from $P \vee Q$:

1.	$P \vee Q$	

2.	| P	
	|---	
3.	| $Q \vee P$	2, $\vee I$
4.	| Q	
	|---	
5.	| $Q \vee P$	4, $\vee I$
6.	$Q \vee P$	1, 2-3, 4-5 SC

Sometimes we argue not from just two cases, but from three or more. The following examples illustrate how to approach such situations.

1.	$P \rightarrow Q$		
2.	$R \rightarrow Q$		
3.	$S \rightarrow Q$		
4.	$(P \vee R) \vee S$		

5.	| S		
	|---		
6.	| $S \rightarrow Q$	3, R	
7.	| Q	5,6 $\rightarrow E$	
8.	| $P \vee R$		
	|---		
9.	| | P		
	| |---		
10.	| | $P \rightarrow Q$	1, R	
11.	| | Q	9,10 $\rightarrow E$	
	|		
12.	| | R		
	| |---		
13.	| | $R \rightarrow Q$	2, R	
14.	| | Q	12,13 $\rightarrow E$	
15.	| Q	8, 9-11, 12-14 SC	
16.	Q	4, 5-7, 8-15 SC	

Example: We derive $P \vee (Q \vee R)$ from $(P \vee Q) \vee R$:

1.	$(P \vee Q) \vee R$	

2.	$P \vee Q$	

3.	P	

4.	$P \vee (Q \vee R)$	4, $\vee I$
5.	Q	

6.	$Q \vee R$	5, $\vee I$
7.	$P \vee (Q \vee R)$	6, $\vee I$
8.	$P \vee (Q \vee R)$	2, 3-4, 5-7 SC
9.	R	

10.	$Q \vee R$	9, $\vee I$
11.	$P \vee (Q \vee R)$	10, $\vee I$
12.	$P \vee (Q \vee R)$	1, 2-8, 9-11 SC

Example: We derive $S \rightarrow R$ from $P \vee Q$, $P \rightarrow R$, $Q \rightarrow (S \rightarrow R)$:

Choosing Implication Introduction as our main attack, we get this:

1.	$P \vee Q$	
2.	$P \rightarrow R$	
3.	$Q \rightarrow (S \rightarrow R)$	

4.	S	

5.	$P \vee Q$	1, R
6.	P	

7.	$P \rightarrow R$	2, R
8.	R	6,7 $\rightarrow E$
9.	Q	

10.	S	4, R
11.	$Q \rightarrow (S \rightarrow R)$	3, R
12.	$S \rightarrow R$	9,11 $\rightarrow E$
13.	R	10,12 $\rightarrow E$
14.	R	5, 6-8, 9-13 SC
15.	$S \rightarrow R$	4-14 $\rightarrow I$

Choosing Separation of Cases as our main attack, we get this:

1.	$P \vee Q$	
2.	$P \rightarrow R$	
3.	$Q \rightarrow (S \rightarrow R)$	

4.	P	

5.	$P \rightarrow R$	2, R
6.	R	4,5 $\rightarrow E$
7.	S	

8.	R	6, R
9.	$S \rightarrow R$	7-8 $\rightarrow I$
10.	Q	

11.	$Q \rightarrow (S \rightarrow R)$	3, R
12.	$S \rightarrow R$	10,11 $\rightarrow E$
13.	$S \rightarrow R$	1, 4-9, 10-12 $\vee E$

(From this last example, with the two different proofs, you should infer that we can have many distinct proofs for the same logical inference.)

Another derived rule is the theorem $(p \vee \neg p)$, sometimes called the Law of Excluded Middle.

LAW OF EXCLUDED MIDDLE (EM)

.	
.	i.e. $p \vee \neg p$ may be written down as a line of a derivation at any point (for any formula p)
.	
$p \vee \neg p$	

Justification: EM is to be understood as an abbreviation of the following schema for a derivation:

1.	| | $\neg(p \vee \neg p)$		
	| |------		
2.	| | | p		
	| | |---		
3.	| | | $p \vee \neg p$		2, $\vee I$
4.	| | | $\neg(p \vee \neg p)$		1, R
5.	| | $\neg p$		2-4 $\neg I$
6.	| | $p \vee \neg p$		5, $\vee I$
7.	$\neg\neg(p \vee \neg p)$		1-6 $\neg I$
8.	$p \vee \neg p$		7, $\neg E$

EM is often handy in arguing by cases:

Example: We derive Q from $P \rightarrow Q$ and $\neg P \rightarrow Q$:

1.	$P \rightarrow Q$		
2.	$\neg P \rightarrow Q$		

3.	$P \vee \neg P$		EM
4.	| P		
	|---		
5.	| $P \rightarrow Q$		1, R
6.	| Q		4,5 $\rightarrow E$
7.	| $\neg P$		
	|---		
8.	| $\neg P \rightarrow Q$		2, R
9.	| Q		7,8 $\rightarrow E$
10.	Q		3, 4-6, 7-9 SC

Another derived rule of inference is this, often called Modus Tollens.

Modus Tollens (MT)

.
.
.
$p \rightarrow q$
.
.
.
$\neg q$
.
.
.
$\neg p$

Here are some more derived rules of inference which you might find handy in proving some of the homework problems.

DeMorgan's Laws: (DM)

$\neg(p \vee q)$	$\neg(p \wedge q)$	$(\neg p \wedge \neg q)$	$\neg p \vee \neg q$
.	.	.	.
.	.	.	.
.	.	.	.
$(\neg p \wedge \neg q)$	$\neg p \vee \neg q$	$\neg(p \vee q)$	$\neg(p \wedge q)$

Divergence (Div)

$p \leftrightarrow q$	$p \leftrightarrow q$
.	.
.	.
.	.
$\neg p$	$\neg q$
.	.
.	.
.	.
$\neg q$	$\neg p$