2.6 Are All the Connectives Necessary?

So far in this chapter we have put together an impressive collection of subroutines which have had the net result of saving us a considerable amount of time and effort. The question is, however, are we beginning to rely on the machine too much? Is it getting beyond our control? Suppose, for example, the machine broke down, even in a minor way; would we still be able to instruct it to check demonstrations by bypassing the faulty unit? To be definite, let's say that there is a fault in the scanner unit which results in the machine's being unable to recognize conditionals when they are input into it.

The scanner unit is part of the mechanism which checks to see if all input formulas are well formed. When functioning properly, here's the way it operates: Suppose the formula

(1) $[P \supset Q]$

is input into the machine. As the tape goes through the scanner unit, messages are sent to the wff unit, which decides whether or not the input formula is well formed. So in this case the scanner unit would transmit the following message: "A left bracket just went by. A variable just went by. A conditional sign just went by. A variable just went by. A right bracket just went by." On receipt of this message the wff unit would then decide that (1) really is a well formed formula. Now if for some reason the scanner unit became unable to recognize a conditional sign, then the message it would transmit to the wff unit would be "A left bracket just went by. A variable just went by. A variable just went by. A right bracket just went by." On the basis of this information the wff unit would reject (1) as being not well formed. The machine would flash its red light and print out

~wff.

Hence this small malfunctioning in the scanner unit results in all conditionals being rejected as being not well formed. Is all lost, or can we find some way to avoid the faulty unit? Fortunately, the General Substitution Principle saves the day. One of the implications of that principle is that if two formulas are equivalent, then one of them may be substituted

for the other whenever it occurs, even if it's a sub-formula of some larger formula. Let's see how that helps us in the case of our faulty scanner unit. We know from Problem 13 of Problem Set 3 that "$\sim P \vee Q$" is equivalent to "$P \Rightarrow Q$". It would seem reasonable that the General Substitution Principle would allow us to transform any demonstration involving conditionals into one with no conditionals, all the original conditionals having been replaced by the appropriate disjunctions. Here's a simple example. If we were asked to show that

(2) $\quad P \Rightarrow \sim R,\ Q \Rightarrow P,\ Q \vdash \sim R$

then we know that our faulty machine would be of no help to us. But perhaps it could help us to show that

(3) $\quad \sim P \vee \sim R,\ \sim Q \vee P,\ Q \vdash \sim R.$

There's just one small problem yet to be surmounted: practically all our demonstrations make use of Modus Ponens, and in order to be able to apply Modus Ponens the machine has to be able to recognize conditionals. It turns out that this is just a programming problem. We'll simply reprogram the machine so that Modus Ponens consists in recognizing $\sim \mathcal{S} \vee \mathcal{T}$ and $\mathcal{S}$ (where $\mathcal{S}$ and $\mathcal{T}$ are well formed formulas) and inferring $\mathcal{T}$. With this modification, let us show (3). First we give the machine the input storage order

Hyp, $\sim P \vee \sim R$, $\sim Q \vee P$, Q.

Then we punch in the following program (remember: Modus Ponens has been reprogrammed):

Pr, Hyp 1

Pr, Hyp 2

Pr, Hyp 3

MP, For 2, For 3

MP, For 1, For 4

and the machine will print out

1) $\sim P \vee \sim R$

2) $\sim Q \vee P$

3) Q

4) P

5) ~R

At this point it is not difficult to see how the General Substitution Principle could be used to transform this demonstration so that it showed (2) if we ever managed to repair the faulty scanner unit. We outline the procedure below (we assume that Modus Ponens has been programmed back to normal again).

	Demonstration Outline	Program Outline
1)	$P \Rightarrow \sim R$	Hyp
2)	$[P \Rightarrow \sim R] \Leftrightarrow [\sim P \vee \sim R]$	Taut
3)	$\sim P \vee \sim R$	MPB or GSP, For 2, For 1 *
4)	$Q \Rightarrow P$	Hyp
5)	$[Q \Rightarrow P] \Leftrightarrow [\sim Q \vee P]$	Taut
6)	$\sim Q \vee P$	MPB or GSP, For 5, For 4 *
7)	Q	Hyp
8)	P	MP, For 4, For 7
9)	~R	MP, For 1, For 8

Notice how this outline has been built up from the previous demonstration. Of course, this demonstration outline does not outline the most efficient way of showing that

$$P \Rightarrow \sim R,\ Q \Rightarrow P,\ Q \vdash \sim R$$

but that is not important. What is important is that this result follows from the fact that we have shown

$$\sim P \vee \sim R,\ \sim Q \vee P,\ Q \vdash \sim R.$$

---

* If we can use MPB in a program outline, we could instead use GSP, but the converse is not true. There are cases in which GSP is used and MPB would not be correct. In this demonstration outline, steps (3) and (6) may be obtained by GSP or by MPB. The MPB should be obvious. GSP may be used because we have "$P \Rightarrow \sim R$" equivalent to "$\sim P \vee \sim R$" and "$\sim P \vee \sim R$" is being substituted for "$P \Rightarrow \sim R$" in formula 1. The second case is similar.

In the demonstration on page 81 of "$S \wedge R$" from "$P \wedge R$" and "$P \Leftrightarrow S$", only GSP may be used at the third step. We don't have either "P" or "S" and as a result can't use MPB with formula 2. However, since we have "$P \Leftrightarrow S$", we can substitute "S" for "P" in the first formula and be sure that the resulting formula is equivalent to the formula in the first step. In cases where MPB and GSP are both correct you only need to list one of them.

Now we can see just how powerful the General Substitution Principle is. It allows us to show the existence of a demonstration simply by producing what might be called an "equivalent" demonstration involving formulas equivalent to the ones in which we are interested.

Let us summarize what we have found so far. We have discovered that if for some reason we are unable to use any formulas which involve the conditional sign, then we can still proceed with our work with demonstrations by making use of the fact that $\mathcal{S} \Rightarrow \mathcal{T}$ and $\sim\mathcal{S} \vee \mathcal{T}$ are equivalent formulas when $\mathcal{S}$ and $\mathcal{T}$ are well formed formulas, and by calling on the General Substitution Principle. So in this sense it is not necessary for our machine to have the capability of recognizing the connective "$\Rightarrow$". In other words, we can handle all problems involving "$\Rightarrow$" using only the connectives "$\sim$", "$\wedge$", and "$\vee$".

This is most interesting, and it raises an important question. In Section 1.6 we found that there are exactly sixteen possible binary operations on the set $\tau = \{t, f\}$ of truth values. That means that there are sixteen possible binary connectives. So far we know about "$\wedge$", "$\vee$", and "$\Rightarrow$" ("$\sim$" is not a <u>binary</u> connective), and we have adopted a fourth symbol "$\Leftrightarrow$", which helps us abbreviate formulas whose truth tables correspond to a fourth one of these sixteen binary operations. These four symbols can fairly faithfully be translated by certain connecting words or phrases in our natural language, English. Who knows what problems we would run into if we tried to apply our propositional calculus to the analysis of arguments in some other language! Perhaps that language would have some connecting words which behave totally differently from those that we have in English. The question is, Would our machine be able to handle any conceivable connective, or would we have to invest a lot of time and money in building into it a whole new set of connectives? The task of discovering the answer to this question is not quite so difficult as it might sound at first, because we know with absolute certainty that there are only twelve possible other connectives. We will simply have to examine them one by one. For each of the twelve we will be looking for some formula of the propositional calculus which has the same truth table. Then we will be able to write formulas equivalent to those which involve the given connective, and so the General Substitution Principle assures us that from then on the situation is manageable.

Exercise 6: Show that it is not necessary to build any new connectives into our machine by forming the remaining twelve truth tables for the possible binary connectives and producing, for each table, a formula of the propositional calculus whose truth table is identical to that table. We will do two examples and leave the other ten for you.

a)

P	Q	
t	t	f
t	f	t
f	t	f
f	f	f

b)

P	Q	
t	t	t
t	f	t
f	t	t
f	f	t

Truth table (a) should immediately remind you of truth table (3) for the conditional; the truth values in the last column are all the same except for one in the second row. However, it's not exactly the same as truth table (3) the "t"s and "f"s in the last column are switched. But that's easy to arrange: you just take the negation. So we have found a well formed formula whose truth table is table (a): "~[P ⇒ Q]".

Truth table (b) is even easier. The last column consists entirely of "t"s, so the formula we are looking for is a tautology. But all tautologies are equivalent to each other, so any tautology will do. In particular, we could use the formula "P ∨ ~P" to produce truth table (b). (Notice that you are not obliged to use both "P" and "Q" in the formulas you produce.)

In the preceding exercise you have shown that all formulas involving any of the sixteen possible binary connectives can be dealt with by producing equivalent formulas that involve only "~", "∧", "∨", and "⇒". But from the earlier part of this section we know that, of these, "⇒" is not really necessary either. So if worse came to worst we could still do the propositional calculus if the only connectives our machine could recognize were "~", "∧", and "∨". Could we reduce that set of connectives even more? Yes. In Problem 10

of Problem Set 3 you showed that "$P \wedge Q$" and "$\sim[\sim P \vee \sim Q]$" are equivalent formulas. Using this fact we could get along without the connective "$\wedge$", and the only connectives our machine would have to be able to recognize would be "$\sim$" and "$\vee$".

So we could do the propositional calculus with only two connectives. Could we do it with just one? Yes! But you have to be careful which connective you pick; neither "$\sim$" nor "$\vee$" will do. It turns out that one of the connectives you looked at in the exercise above does the trick. This connective is called the Sheffer stroke (or alternative denial). It is denoted by "$|$", and if $\mathcal{S}$ and $\mathcal{T}$ are well formed formulas, its truth table is the following:

$\mathcal{S}$	$\mathcal{T}$	$\mathcal{S} \mid \mathcal{T}$
t	t	f
t	f	t
f	t	t
f	f	t

Exercise 7: Show that there is a formula involving no connectives except the Sheffer stroke that is equivalent to "$\sim P$".

Exercise 8: Show that there is a formula involving no connectives except the Sheffer stroke that is equivalent to "$P \vee Q$".

Since we already know that every conceivable formula of the propositional calculus is equivalent to a formula that involves no connectives other than "$\sim$" and "$\vee$", it follows from Exercises 7 and 8 that every formula of the propositional calculus is equivalent to a formula that contains no connectives except the Sheffer stroke. There is a nice English word that is appropriate here: redundant. Something is redundant if it is not really needed. There is a great deal of redundancy in the English language. There are many pairs of English words that are so close in meaning that we could express ourselves quite satisfactorily if some of these words were deleted from the language.

Our alphabet contains redundancies. Try the following experiment. Take a piece of paper that you cannot see through. Open any book and place the paper so that its edge

runs through the middle of a line of print and enables you to see only the top half of each letter. Can you still read the sentence? If so, then the bottom halves of those letters aren't really needed. Try it again covering up the top half of each letter. What should we conclude from this? Is redundancy an evil that should be eliminated? Should we print books using only the top halves of the letters of the alphabet? If we did, we could get the same amount of information on half as much paper. Should we formulate the propositional calculus in terms of the Sheffer stroke? This would probably not be a good idea. Let us illustrate why.

Try the earlier experiment of covering the bottom half of a line of print but this time slide the paper up to cover more than the bottom half. You will most likely still be able to read it even when you can only see a small part of the top of each letter. But it is fairly likely that you have to work harder and read more slowly. Did you pause to ask yourself questions such as: Is that the top of an "s" or a "c"? Mr. Sheffer's stroke poses the same problem. We have shown above that we could do the propositional calculus with it but we would have to work harder. The connectives we have chosen are more natural and easier to deal with.

Perhaps we have overworked this discussion so let us sum up two points. Through the notion of equivalence and with the help of the General Substitution Principle we have shown that the propositional calculus, as we have formulated it, contains certain redundancies. This fact provides us with options in formulating the calculus. While it is true that if two formulas are equivalent we could do without one of them, there is another side to the redundancy coin. If two formulas are equivalent these formulas tell us two different ways to think about some problem. This is of great value. In fact, success in solving many a problem has come only after finding out which of the many equivalent statements of the problem provides a manageable way of thinking about it.

See Appendix II for material that will help you review for the test on the first part of Chapter 2.

2.7 The Deduction Theorem

With the subroutines we have introduced, and others that we can imagine introducing, we see that our machine is becoming of greater value in our attempt to analyze mathematical arguments. If this attempt is successful, that is, if we succeed in showing that all, or even many, mathematical arguments can be checked by our machine, then that in itself will be a remarkable fact because our machine is really incredibly simple. As things stand, however, our machine has one drawback: we have to know any given demonstration beforehand. We have to know what it is that we want the machine to check so that we can feed it the correct instructions when we program it to check a demonstration. It would be so much more useful to have a machine that actually generated demonstrations.

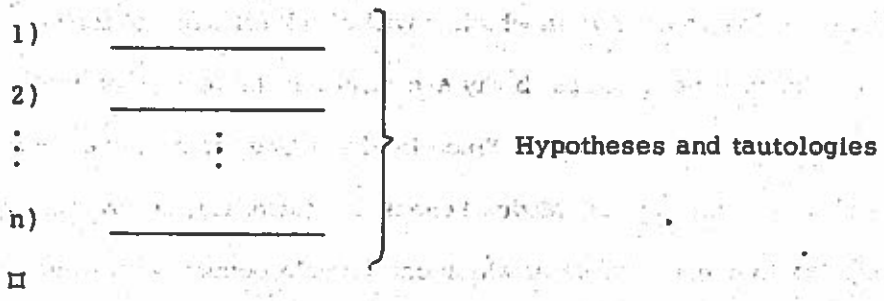
Looking behind the subroutines we see that all our machine really does is to compare pairs of formulas to see if they are compatible for Modus Ponens, and upon finding that two formulas are compatible it applies the rule. This observation suggests the possibility of a completely different type of machine, a machine that searches out all the logical consequences of a given set of hypotheses. To see how this is possible let us look more closely at what our present machine does. Since it was our first machine let us call it "Model A".

To program Model A to check an argument, or more correctly, to program it to print a demonstration, we first give the machine input storage orders. Two kinds of formulas are stored, hypotheses and tautologies. We then give the machine a program which is basically a set of print orders. There are two kinds of print orders, orders to print stored formulas, either hypotheses or tautologies; and MP-orders which are orders to print the consequent of a specified conditional, provided another specified formula is the antecedent of that conditional. We can dispose of the first type of print order by designing our machine so that, as soon as you punch a button indicating that you have finished giving it input storage orders, it immediately prints all the hypotheses and tautologies that it has stored.

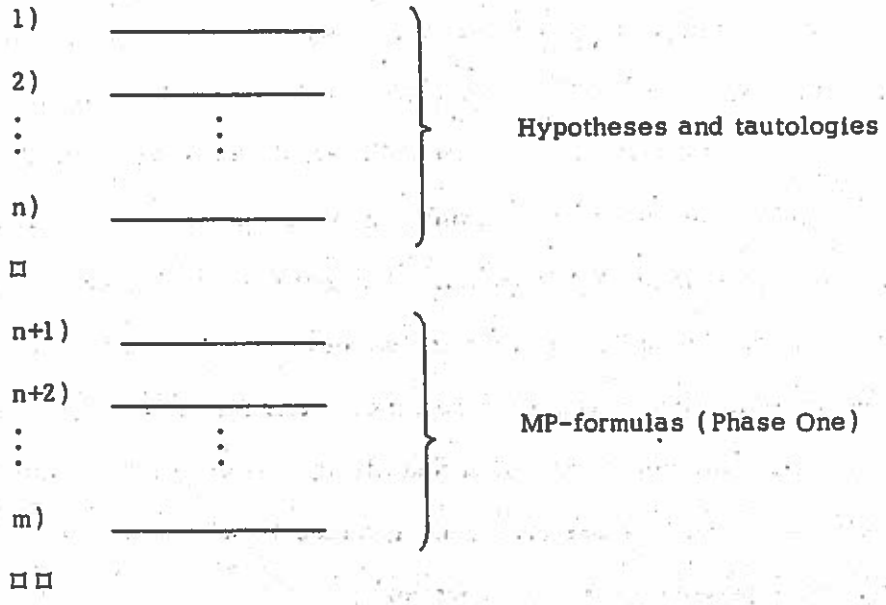
At this point you should pause and reflect on this modification of our machine. Imagine any demonstration. Suppose that we move all the hypotheses and tautologies up in the sequence of formulas so that they come first. Will this rearrangement of the formulas still

be a demonstration? We hope it is clear that the answer is "yes". Every demonstration has three kinds of formulas: hypotheses, tautologies, and formulas obtained by applying Modus Ponens. Let us call formulas of this third type, "MP-formulas". We have now agreed that the formulas of a demonstration can be rearranged so that the hypotheses and tautologies come first and then come all the MP-formulas. This fact will be of interest to us shortly.

To decide on the design of our new machine let us discuss some possibilities and then come to an agreement about what must be done. Consider the following plan. Our new machine, at the punch of a button indicating that all input storage orders have been given, immediately prints all input formulas (i.e., all hypotheses and tautologies). It then prints a special mark whose purpose we will soon explain.

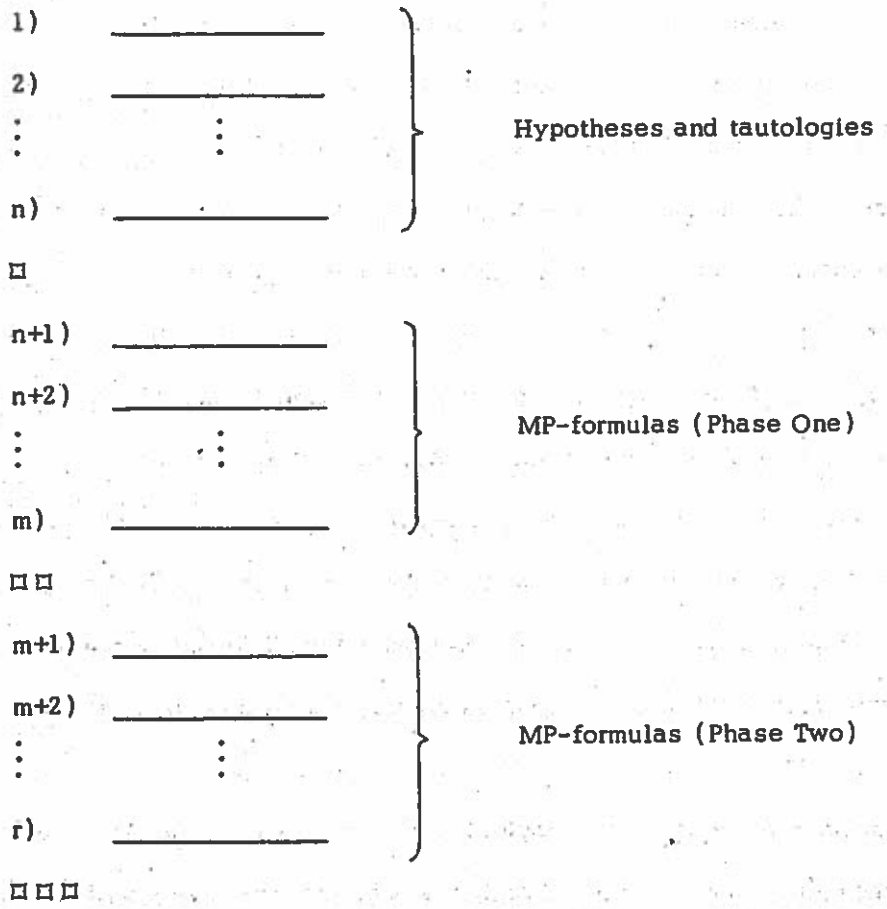


The old Model A machine responds to an MP-order only when told specifically which two formulas it should consider. But suppose we design our new machine with a built-in set of MP-orders of the following kind. First the machine compares For 1 and For 2 to see if they are compatible for Modus Ponens. Let it be programmed to consider both possible orders of listing the major and minor premises. If a compatible combination is found, the machine applies Modus Ponens and prints the result. Clearly this result is a logical consequence of the given hypotheses. The machine then continues by comparing For 1 with For 3, then For 1 with For 4, etc., until it reaches the special mark, whereupon it commences a new sequence of comparisons, comparing For 2 with For 3, then For 2 with For 4, etc., until it has compared all formulas before the mark two by two and printed all results of applying Modus Ponens to compatible pairs of formulas. This completes Phase One and the machine places a second mark after the last formula it has printed.



The machine next begins Phase Two in which it will check all pairs of formulas before the second mark for Modus Ponens compatibility and print out the results of applying Modus Ponens to compatible pairs of formulas. Since in Phase One the machine checked each pair of formulas before the first mark for Modus Ponens compatibility, it now only has two sorts of pairs of formulas to check: those of which one formula occurs before the first mark and the other occurs between the first and second mark; and those pairs of which both formulas occur between the two marks. So our machine first of all compares each of the formulas before the first mark with each of the formulas between the two marks, and then treats the formulas between the marks in the same way it treated the formulas before the first mark in Phase One.

Having completed Phase Two the machine places a third mark after the last formula printed and begins Phase Three, namely, to compare all formulas before the third mark.



Phase Three proceeds in much the same way as Phase Two with the exception that we now think of the formulas as either being before the second mark or between the second and third marks, That is, from Phase Three on we can forget about the first mark. Phases Four, Five, and so on follow in an analogous fashion. In each phase we can forget about all but the last two marks.

Again let us pause and reflect on what our machine is doing. You will probably have realized that there is quite a good possibility of our new machine continuing working, printing out MP-formulas forever. At each phase there is a larger set of formulas that are being compared pairwise for Modus Ponens compatibility, and so there is more opportunity for compatible pairs to be found, thus causing some more MP-formulas to be printed. It should be clear that in Phase One the machine tests all pairs of formulas before the first mark for Modus Ponens compatibility. It should also be clear that by the end of Phase Two it will have tested all pairs of formulas before the second mark for Modus Ponens compatibility.

We wanted our machine to print out logical consequences of the input hypotheses. Can we be sure that every formula the machine prints is a logical consequence of those hypotheses? By definition a formula is a logical consequence of a set of hypotheses provided there is a demonstration of that formula from these hypotheses and no others. How can we be sure that such a demonstration exists for each formula the machine prints? That is easy. Think about any particular formula that the machine prints. Imagine stopping the machine as soon as it prints this formula. Then the machine printout is a demonstration of the last formula printed! It may not be the shortest demonstration of that formula and it may contain lots of formulas that are not essential to the demonstration, but it is a demonstration of this particular formula. It is a demonstration because every formula in the printout is either a hypothesis, or a tautology, or a formula obtained by Modus Ponens from a pair of formulas that appear earlier on the list. The definition of a demonstration does not say that it must be neat, tidy, and as short as possible.

So our machine is generating logical consequences of the given hypotheses, but will it generate them all? With a little reflection we see that we have a problem. If we only concern ourselves with finite collections of hypotheses, there is no problem in giving them as input data and no problem for the machine to print all of these hypotheses. There is a problem however, with the tautologies for there are infinitely many of them. Even if we had some way of feeding all tautologies into the machine, it is currently so designed that it must print them all before it makes the first mark. The machine will be printing tautologies until Doomsday and will never reach the first mark.

There is a way around this difficulty that is clever yet simple. Suppose that our hypotheses are $G_1, \ldots, G_n$ and that somehow we have a tautology generator that feeds the tautologies into the machine in some order[*]

---

[*] Although there are an infinite number of tautologies, it is still possible to number then in some order. In order to be convinced of this, however, you will have to wait until later in your mathematical studies. A numerical example of this type of situation is: the elements of $\mathbb{Z}$, the set of integers, can be numbered in order $z_1, z_2, z_3, \ldots$ For example, we could let $z_1 = 0$, $z_2 = 1$, $z_3 = -1$, $z_4 = 2$, $z_5 = -2$, $z_6 = 3$, $z_7 = -3$, ... On the other hand, the elements of $\mathbb{R}$, the set of real numbers, cannot be numbered in order.

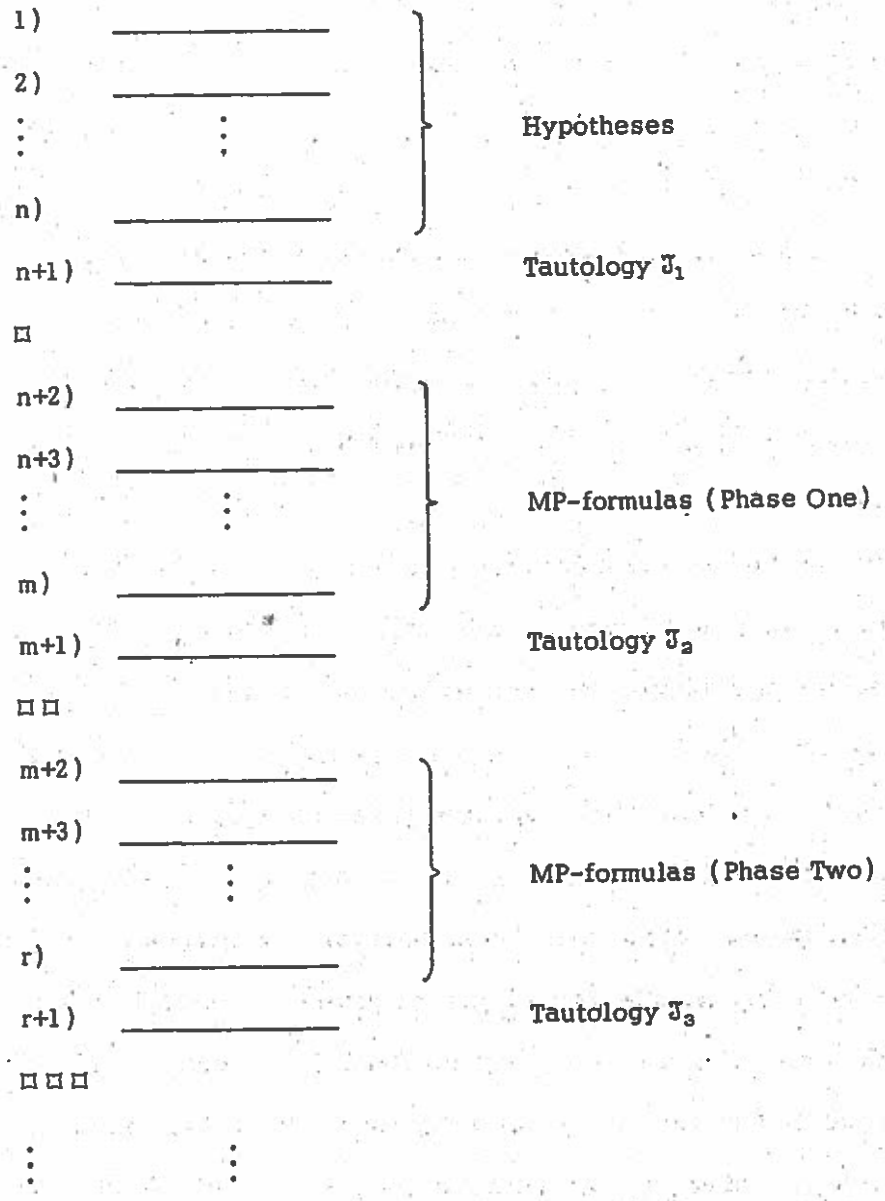
$\mathfrak{I}_1, \mathfrak{I}_2, \ldots$

so that each tautology will be stored in the machine at some point in time. Then we modify the machine slightly so that it prints

$G_1, \ldots, G_n, \mathfrak{I}_1$

and then makes its first mark. It compares all pairs of formulas before the first mark for Modus Ponens compatibility and prints the results of applying Modus Ponens to compatible pairs. The machine then prints $\mathfrak{I}_2$, makes its second mark, and goes into Phase Two. On completing Phase Two it prints $\mathfrak{I}_3$, makes its third mark, and goes into Phase Three, etc. (See page 97.)

Again it is clear that every formula the machine prints will be a logical consequence of the hypotheses. But, more important, this time we have the added feature that any given logical consequence of the hypotheses will be printed at some moment. That is, given any formula that is a logical consequence of $G_1, \ldots, G_n$, our machine will eventually print it. How do we know? Well, think about any formula $\mathfrak{G}$ that is a consequence of $G_1, \ldots, G_n$. There must be a demonstration of $\mathfrak{G}$ from $G_1, \ldots, G_n$, and indeed one in which all hypotheses precede all MP-formulas. Just to simplify our discussion, let us assume that this demonstration of $\mathfrak{G}$ contains no tautologies, but that it contains k MP-formulas, say $C_1, \ldots, C_k$, that occur in that order. Then C_k is the formula $\mathfrak{G}$ itself. Consider the first MP-formula, C_1. Since it is the first MP-formula it must follow by Modus Ponens from two of the hypotheses. Since the machine printed all the hypotheses before the first mark and since C_1 follows by Modus Ponens from two of the hypotheses, the machine will print C_1 somewhere between the first and second marks. Similarly, C_2 follows by Modus Ponens from a pair of formulas taken from the set of formulas consisting of all the hypotheses and C_1, and hence the machine will print C_2 somewhere between mark 1 and mark 3. Continuing in this way we see that $\mathfrak{G}$, which is the kth MP-formula in the demonstration, will be printed somewhere between the first and (k+1)th mark. In summary, if $\mathfrak{G}$ is the kth MP-formula in its demonstration then it will be printed by the time the machine completes Phase k.



If a demonstration of $\mathfrak{G}$ involves tautologies, we know that these tautologies will eventually be fed into the machine and printed. By the same argument as before we see that if $\mathfrak{G}$ is the kth MP-formula in this demonstration, then the machine will print $\mathfrak{G}$ within k phases after it has printed the last tautology that is needed. It may in fact print $\mathfrak{G}$ sooner; it could even print it just one phase after it has printed the last tautology the demonstration requires. Can you explain why?

Let us call the machine that is modified in the way we have been discussing, "Model T". Before our Model T is operational there is one bug we must remove. We must

design a tautology generator. There are a variety of ways in which this can be done. Let us, however, set the problem aside to reflect on the use of the Model T should it ever be completed. It will surely be an extraordinary machine. If we give it an input storage order

$$\text{Hyp}, G_1, G_2, \ldots, G_n .$$

and push the print button, Model T begins printing out all of the logical consequences of the hypotheses $G_1, \ldots, G_n$.

If we succeed in getting Model T operational, should we then retire all professional mathematicians and assign the mathematical journals the task of publishing the printouts of Model T? Surely not. Most of what Model T will print out is of no interest to anyone. The situation is much like the following.

If a whole tribe of monkeys is taught how to punch the keys of a typewriter and each monkey is kept busily at work at the keyboard, then on statistical grounds it can be argued that eventually, just by sheer accident, one of these monkeys will type a page of Shakespeare. However, if you try this experiment don't hold your breath while you wait. We (and the monkeys) will almost certainly all die of old age long before a page appears that is of the remotest interest to anyone. Alas, the same is true of our Model T. This machine, like the monkeys, just does not have good taste. Only humans know what is interesting and useful and important. While we may be interested, in some sense, to know what all the logical consequences of a given set of hypotheses are, we are not interested in each particular consequence.

Perhaps the way to use Model T is not just to sit and watch the printout hoping that something interesting will appear, but rather to search for interesting conjectures and try to get the machine to tell us whether our conjectures are correct. No, that won't work either. If we have some conjecture, that is, some formula that we think is a consequence of certain hypotheses, we could feed the hypotheses into the machine, press the print button and wait, and wait, and wait. Even if our formula is a consequence of the given hypotheses, we may have to wait centuries for the machine to confirm this by printing the formula. Furthermore,

if our formula is not a consequence of the hypotheses, the machine will never print it, but how are we to know that it will never be printed? This raises an interesting question. Can we decide in advance whether the machine will ever print a given formula? We discuss this question in the next few pages and provide an answer after we have introduced a very important principle about demonstrations.

There are a few observations about our machines that should be made at this point. The machines have been very helpful to us in our study of the propositional calculus by compelling us to think very carefully about certain issues. Through the use of subroutines the machines can be designed to do certain "mechanical" things and hence save us time by removing the necessity to write out a demonstration in full; an outline is perfectly adequate for most purposes. However, the machines do only what we tell them to do. They cannot think and hence when we are confronted with a problem that requires thought and imagination, the machine cannot provide that thought and imagination. We must do our own thinking. Faced with a problem that requires new procedures, or new insight, or new understanding, we must discover the procedures, gain the insight, acquire the understanding.

Could we find a procedure that will enable us to decide in advance whether our new machine (Model T) will ever print a given formula? We know that the machine will print a formula iff that formula has a demonstration. Thus far we have had only one method for determining whether a formula has a demonstration: we simply hunt until we find one. All the problems we have examined up to this point have been carefully selected so that with a little work and thought a demonstration can be found. Suppose we were asked to show that $P \vee Q \vdash Q$. In fact no such demonstration exists, but how are we to know this? If we search in vain, all we can report is that we have failed to find a demonstration. In general, failure to find a demonstration does not mean that the desired demonstration does not exist. Wouldn't it be nice to have some simple test that we could apply to find out whether a given formula $\mathcal{B}$ is a logical consequence of hypotheses $G_1, \ldots, G_n$? If we apply the test and discover that the answer is "no", then we know there is no point in looking for a demonstration.

If we apply the test and the answer is "yes", then we can look for a demonstration confident that we are looking for something that does exist.

Is it reasonable to ask for such a test? Let us begin with a very simple case. Can we find a test that will tell us whether a formula $\mathcal{B}$ requires any hypotheses at all in its demonstration, i.e., whether $\vdash \mathcal{B}$? We have already noted that if $\mathcal{B}$ is a tautology, then $\mathcal{B}$ is a demonstration of itself with no hypotheses. That is,

If $\mathcal{B}$ is a tautology, then $\vdash \mathcal{B}$.

Suppose that there is a demonstration of $\mathcal{B}$ with no hypotheses. Does it follow that $\mathcal{B}$ is a tautology? The answer is "yes". Consider any demonstration of $\mathcal{B}$ that is hypothesis-free. Such a demonstration will be a sequence of formulas ending with $\mathcal{B}$, for which each individual formula of the sequence is a tautology or follows by Modus Ponens from a pair of formulas that appear earlier in the sequence. Suppose that $\mathcal{S}$ and $\mathcal{S} \Rightarrow \mathcal{T}$ are tautologies that occur in the sequence. It then follows that $\mathcal{T}$ must also be a tautology. (Why?) That is, a formula obtained by Modus Ponens from two tautologies is also a tautology. From this observation it follows that every formula in the given demonstration must be a tautology, for suppose that some formula of the sequence is not a tautology. Then there must be a first formula that is not a tautology. But this first nontautology must follow by Modus Ponens from two formulas that are tautologies. (Why?) Since this contradicts what we concluded at the beginning of this paragraph, it must be the case that every formula in the demonstration including $\mathcal{B}$ which is the last formula, is a tautology.

We have just proved the following result.

Principle of Hypothesis-Free Demonstration: $\vdash \mathcal{B}$ iff $\mathcal{B}$ is a tautology.

Using truth tables we can determine, in a purely mechanical way, whether a given formula $\mathcal{B}$ is a tautology. Consequently, using the principle just stated, we can determine, in a purely mechanical way whether $\mathcal{B}$ has a hypothesis-free demonstration. Clearly, once we have discovered, using truth tables, that $\mathcal{B}$ is a tautology, then we know one hypothesis-

free demonstration of $\mathcal{B}$, namely the one-step demonstration consisting simply of $\mathcal{B}$ itself. Conversely, for some formulas it may be easier to produce a hypothesis-free demonstration than to complete a truth table. For such formulas we now have another way (besides using truth tables) to prove that they are tautologies—thanks to the Principle of Hypothesis-Free Demonstrations.

But now let us return to the more general problem of finding a test that will tell us whether a given formula is a logical consequence of a given set of hypotheses. One such test is provided by the following result.

Deduction Theorem*:

Suppose $G_1, \ldots, G_n, \mathcal{B}, C$ $(n \in N)$ are well formed formulas, then

i) $\mathcal{B} \vdash C$ iff $\vdash \mathcal{B} \Rightarrow C$,

and

ii) $G_1, \ldots, G_n, \mathcal{B} \vdash C$ iff $G_1, \ldots, G_n \vdash \mathcal{B} \Rightarrow C$.

Notice that part (i) complements part (ii) in the sense that part (ii) takes care of the situations in which there are a certain number of hypotheses in addition to the one which "moves" to the other side of the "$\vdash$". On the other hand, part (i) deals with the situation in which there are no such additional hypotheses. Let us begin to gain an understanding of what this result is saying by considering part (i).

We know that $\vdash \mathcal{B} \Rightarrow C$ iff $\mathcal{B} \Rightarrow C$ is a tautology. Therefore, the Deduction Theorem tells us that there is a demonstration of C from $\mathcal{B}$ iff $\mathcal{B} \Rightarrow C$ is a tautology. For example, since "$[P \wedge Q] \Rightarrow Q$" is a tautology we know that $P \wedge Q \vdash Q$; since "$[P \vee Q] \Rightarrow Q$" is not a tautology we know that "Q" is not a logical consequence of "$P \vee Q$".

---

* In order to stay in line with the names we have given similar results on the foregoing pages we should perhaps have called this result "The Deduction Principle". However, it is used so frequently in mathematical arguments that we thought it best to use the name by which it is most commonly known. For the time being you may take "theorem" as being synonymous with "principle". In fact, the word "theorem" is being used in a technical sense in this instance, and that technical meaning is discussed at a later time in the EM series.

If we have more than one hypothesis, we can apply the Deduction Theorem more than once:

$$P, Q \vdash P \wedge Q \text{ iff } P \vdash Q \Rightarrow [P \wedge Q]$$

$$\text{iff } \vdash P \Rightarrow [Q \Rightarrow [P \wedge Q]].$$

Since "$P \Rightarrow [Q \Rightarrow [P \wedge Q]]$" is a tautology (check this), "$P \wedge Q$" is a logical consequence of "P" and "Q". Of course we proved this earlier. From this example we see that the Deduction Theorem provides us with a fail-safe way of discovering a demonstration (if it exists) of a given result. One writes out all the hypotheses in a long string to the left of the desired formula, places "$\Rightarrow$" followed by "[" between each pair of consecutive formulas, and adds sufficient right hand brackets at the end to match the inserted left hand brackets. Then one tests the resulting formula to see if it is a tautology. If it is not, repeated use of the Deduction Theorem tells us that the given formula is not a logical consequence of the given hypotheses. On the other hand, if it is a tautology, then a demonstration is obtained by writing down this tautology, then listing the hypotheses and repeatedly applying Modus Ponens. However, this method runs counter to the spirit of what we are trying to do. The Deduction Theorem was introduced to save us some work, but the above method requires us to perform a very tedious check to see whether a certain (often quite complicated) formula is a tautology[*]. A more inspired use of the Deduction Theorem does in fact result in a great reduction of labor. Here is an example of such a use.

Suppose that we wish to show that

(1) $$S, [S \wedge P] \Rightarrow Q \vdash P \Rightarrow Q.$$

At first glance it is not clear how we should proceed in order to produce the demonstration required to show (1). The Deduction Theorem however assures us that in order to show that such a demonstration exists it is sufficient to show the existence of a demonstration of a different result:

---

[*] This method is guaranteed to produce a demonstration (if it exists). But it all turns on a certain formula being a tautology. Hence if you decide to use this method to produce a demonstration in a given instance, you will be obliged to furnish a truth table verification that the appropriate formula is a tautology.

(2) $S, [S \wedge P] \Rightarrow Q, P \vdash Q.$

But it is obvious how we would go about producing the demonstration referred to in (2). We would infer " $S \wedge P$ " from "S" and "P" by Conjunctive Inference, and then infer "Q" from " $[S \wedge P] \Rightarrow Q$ " and " $[S \wedge P]$ " by Modus Ponens.

We hope these few examples (particularly the last one) have convinced you of the value of the Deduction Theorem. Instead of trying to establish the validity of the Deduction Theorem in its most general form we are going to deal only with one special case, that of showing that (1) holds if and only if[*] (2) holds. We proceed in this way since this special case illustrates all of the ideas in the general argument so well that if you understand the special case you will have a fair understanding of the general case. Our procedure will be the following: First we will show that if (1) holds then (2) must also hold, and second, we will show that if (2) holds then (1) must hold. From (1) we know that there exists a demonstration of " $P \Rightarrow Q$ " from "S" and " $[S \wedge P] \Rightarrow Q$ ", so consider one such demonstration, for example:

	Demonstration	Program Outline[◇]
1)	S	Hyp
2)	$[S \wedge P] \Rightarrow Q$	Hyp
3)	$S \Rightarrow [[[S \wedge P] \Rightarrow Q] \Rightarrow [P \Rightarrow Q]]$	Taut[□]
4)	$[[S \wedge P] \Rightarrow Q] \Rightarrow [P \Rightarrow Q]$	MP, For 3, For 1
5)	$P \Rightarrow Q$	MP, For 4, For 2

If at the end of this demonstration we add the two formulas

6)	P	Hyp
7)	Q	MP, For 5, For 6

---

[*] The English phrase "A if and only if B" should be interpreted as meaning "if A, then B; and if B, then A". Compare this with the situation in the propositional calculus where we are using " $P \Leftrightarrow Q$ " as an abbreviation for " $[P \Rightarrow Q] \wedge [Q \Rightarrow P]$ ".

◇ From this point on we are back with our Model A machine. This is because the introduction of the Deduction Theorem makes it possible (in a way which we explain below) for us to use the Model A machine to do the sort of job we were trying to get done when we designed the Model T.

□ Don't forget to check that this really is a tautology.

we obtain a demonstration of "Q" from "S", " $[S \wedge P] \Rightarrow Q$ ", and "P".

You will have noticed how easy it was to extend the first demonstration to the second. In fact, it is always this easy, even in the general case. The part of the Deduction Theorem that says

$$\text{If } G_1, \ldots, G_n \vdash \mathcal{B} \Rightarrow C \text{ then } G_1, \ldots, G_n, \mathcal{B} \vdash C$$

may be justified in general simply by noting that any k-formula demonstration of $\mathcal{B} \Rightarrow C$ from $G_1, \ldots, G_n$ may be transformed into a (k+2)-formula demonstration of C from $G_1, \ldots, G_n, \mathcal{B}$ simply by adding the steps

k+1)	$\mathcal{B}$	Hyp
k+2)	C	MP, For k, For (k+1).

"Going the other way", that is, showing that

$$\text{If } G_1, \ldots, G_n, \mathcal{B} \vdash C \text{ then } G_1, \ldots, G_n \vdash \mathcal{B} \Rightarrow C$$

is much less trivial. We return to our special case.

We need to show that if (2) holds, then (1) holds. Now, if (2) holds then there exists a demonstration of "Q" from the hypotheses "S", " $[S \wedge P] \Rightarrow Q$ " and "P". Consider any such demonstration, for example:

	Demonstration	Program Outline
1)	S	Hyp
2)	$[S \wedge P] \Rightarrow Q$	Hyp
3)	P	Hyp
4)	$S \Rightarrow [P \Rightarrow [S \wedge P]]$	Taut
5)	$P \Rightarrow [S \wedge P]$	MP, For 4, For 1
6)	$S \wedge P$	MP, For 5, For 3
7)	Q	MP, For 2, For 6

We now show how to convert this demonstration into a demonstration of " $P \Rightarrow Q$ " from the hypotheses "S" and " $[S \wedge P] \Rightarrow Q$ ". The first step is to form a conditional with antecedent "P" from each of the seven formulas in the given demonstration.

1′) $P \Rightarrow S$

2′) $P \Rightarrow [[S \wedge P] \Rightarrow Q]$

3′) $P \Rightarrow P$

4′) $P \Rightarrow [S \Rightarrow [P \Rightarrow [S \wedge P]]]$

5′) $P \Rightarrow [P \Rightarrow [S \wedge P]]$

6′) $P \Rightarrow [S \wedge P]$

7′) $P \Rightarrow Q$

These seven formulas do <u>not</u> form the desired demonstration. But by adding appropriate formulas to the sequence we will obtain that demonstration. We divide the formulas (1)-(7) into the following four cases:

Case 1: the formula is a hypothesis different from "P";

Case 2: the formula is the hypothesis "P";

Case 3: the formula is a tautology;

Case 4: the formula follows by Modus Ponens from formulas that occur earlier in the sequence.

In formula (1), "S" is a hypothesis different from "P". In such a case we add two formulas before formula (1′):

S

$S \Rightarrow [P \Rightarrow S]$

1′) $P \Rightarrow S$

Here the first formula is a hypothesis (of the demonstration we are trying to construct), the second is a tautology, and formula (1′) then follows from the first two by Modus Ponens.

Formula (2) is again a hypothesis different from "P", so we add two steps between formulas (1′) and (2′):

$[S \wedge P] \Rightarrow Q$

$[[S \wedge P] \Rightarrow Q] \Rightarrow [P \Rightarrow [[S \wedge P] \Rightarrow Q]]$

2′) $P \Rightarrow [[S \wedge P] \Rightarrow Q]$

Again, the first of these formulas is a hypothesis (of the demonstration we are constructing), the second is a tautology, and formula (2′) follows from these two by Modus Ponens.

Formula (3) is the hypothesis "P". Since "$P \Rightarrow P$" is a tautology we need no formulas between (2′) and (3′). So far we have the following steps in the demonstration we are constructing:

S

$S \Rightarrow [P \Rightarrow S]$

1′) $P \Rightarrow S$

$[S \wedge P] \Rightarrow Q$

$[[S \wedge P] \Rightarrow Q] \Rightarrow [P \Rightarrow [[S \wedge P] \Rightarrow Q]]$

2′) $P \Rightarrow [[S \wedge P] \Rightarrow Q]$

3′) $P \Rightarrow P$

Formula (4) is a tautology. Since any conditional with a tautology as a consequent is also a tautology (see Problem 24 of Problem Set 3), it follows that (4′) is a tautology. Consequently we need add no new formulas between (3′) and (4′).

4′) $P \Rightarrow [S \Rightarrow [P \Rightarrow [S \wedge P]]]$

Formula (5) follows by Modus Ponens from (4) and (1). We will add two formulas before (5′) that will enable us to obtain (5′) by Modus Ponens using (4′) and (1′).

The easy way to remember which formulas to add is by applying part of what is known as the Self-distributive Law of the Conditional. In Problem 6 of Problem Set 4 you showed that

$$[P \Rightarrow [Q \Rightarrow R]] \Leftrightarrow [[P \Rightarrow Q] \Rightarrow [P \Rightarrow R]]$$

is a tautology. Then, from Problem 23 of Problem Set 3, we know that

$$[P \Rightarrow [Q \Rightarrow R]] \Rightarrow [[P \Rightarrow Q] \Rightarrow [P \Rightarrow R]]$$

is also a tautology.

As an instance of this tautology with

P: P

Q: S

and R: $P \Rightarrow [S \wedge P]$

we have

(*) $[P \Rightarrow [S \Rightarrow [P \Rightarrow [S \wedge P]]]] \Rightarrow [[P \Rightarrow S] \Rightarrow [P \Rightarrow [P \Rightarrow [S \wedge P]]]]$

which by the Tautology Principle is also a tautology. Not only is it a tautology, but it has (4′) as its antecedent. Consequently, from (*) and (4′) we have, by Modus Ponens,

$$[P \Rightarrow S] \Rightarrow [P \Rightarrow [P \Rightarrow [S \wedge P]]]$$

which has (1′) as its antecedent. Therefore, by Modus Ponens again, we have

(5′) $P \Rightarrow [P \Rightarrow [S \wedge P]]$.

Since (6) follows from (5) and (3) by Modus Ponens we apply the partial distributive law for the conditional to (5′) to discover the two formulas we should add before (6′). Then finally, since (7) follows from (2) and (6) by Modus Ponens, we apply the partial distributive law for the conditional to (2′) to discover the two formulas we should add before (7′).

When we are finished it will look like the following. For your convenience we renumber the steps and provide an outline of a Model A machine program that would produce a machine check to show that we really have come up with a demonstration.

	Demonstration	Program Outline
1)	S	Hyp
2)	$S \Rightarrow [P \Rightarrow S]$	Taut
3) = (1′)	$P \Rightarrow S$	MP, For 2, For 1
4)	$[S \wedge P] \Rightarrow Q$	Hyp
5)	$[[S \wedge P] \Rightarrow Q] \Rightarrow [P \Rightarrow [[S \wedge P] \Rightarrow Q]]$	Taut
6) = (2′)	$P \Rightarrow [[S \wedge P] \Rightarrow Q]$	MP, For 5, For 4
7) = (3′)	$P \Rightarrow P$	Taut
8) = (4′)	$P \Rightarrow [S \Rightarrow [P \Rightarrow [S \wedge P]]]$	Taut
9)	$[P \Rightarrow [S \Rightarrow [P \Rightarrow [S \wedge P]]]] \Rightarrow [[P \Rightarrow S] \Rightarrow [P \Rightarrow [P \Rightarrow [S \wedge P]]]]$	Taut
10)	$[P \Rightarrow S] \Rightarrow [P \Rightarrow [P \Rightarrow [S \wedge P]]]$	MP, For 9, For 8

11) = (5′)	$P \Rightarrow [P \Rightarrow [S \wedge P]]$	MP, For 10, For 3
12)	$[P \Rightarrow [P \Rightarrow [S \wedge P]]] \Rightarrow$ $[[P \Rightarrow P] \Rightarrow [P \Rightarrow [S \wedge P]]]$	Taut
13)	$[P \Rightarrow P] \Rightarrow [P \Rightarrow [S \wedge P]]$	MP, For 12, For 11
14) = (6′)	$P \Rightarrow [S \wedge P]$	MP, For 13, For 7
15)	$[P \Rightarrow [[S \wedge P] \Rightarrow Q]] \Rightarrow$ $[[P \Rightarrow [S \wedge P]] \Rightarrow [P \Rightarrow Q]]$	Taut
16)	$[P \Rightarrow [S \wedge P]] \Rightarrow [P \Rightarrow Q]$	MP, For 15, For 6
17) = (7′)	$P \Rightarrow Q$	MP, For 16, For 14

Although some of the formulas may be long the procedure we have described is really very simple. There are four cases to consider but in only two cases do we add new formulas:

(i) If G is a hypothesis different from the special hypothesis S, we insert the hypothesis G and the tautology $G \Rightarrow [S \Rightarrow G]$ immediately before $S \Rightarrow G$.

(ii) If G follows from $\mathcal{B}$ and $\mathcal{B} \Rightarrow G$ by Modus Ponens, we apply the partial distributive law for the conditional to

$$S \Rightarrow [\mathcal{B} \Rightarrow G]$$

and insert the two steps

$$[S \Rightarrow [\mathcal{B} \Rightarrow G]] \Rightarrow [[S \Rightarrow \mathcal{B}] \Rightarrow [S \Rightarrow G]]$$

$$[S \Rightarrow \mathcal{B}] \Rightarrow [S \Rightarrow G]$$

immediately before $S \Rightarrow G$.

At this point you should pause and consider why we may be sure that the procedure described above will always produce a demonstration of the desired result. How can we be sure that the correct Modus Ponens-compatible pairs of formulas will show up as a result of applying this procedure? Let us consider another example.

Suppose we are asked to show that "$[P \wedge R] \Rightarrow Q$" is a logical consequence of "$R \Rightarrow Q$". We will do this by first showing that "Q" is a logical consequence of "$R \Rightarrow Q$"

and " P ∧ R", and then transforming this demonstration into one of " [P ∧ R] ⇒ Q " from " R ⇒ Q". This transformation is done in a way that is so straightforward that we can provide our Model A machine with a new subroutine that enables it to do all the work at a single command. Here is how we proceed.

The input storage order looks like this

Hyp, R ⇒ Q, Dis P ∧ R.

The "Dis" is a signal to the machine that it is about to receive a program whose instructions it is to carry out, storing the resulting information until a special command is received. We then feed into the machine a program for a demonstration of "Q" from the hypotheses " R ⇒ Q " and " P ∧ R". For example,

	Demonstration*	Program Outline
1)	R ⇒ Q	Hyp
2)	P ∧ R	Dis Hyp
3)	[P ∧ R] ⇒ R	Taut
4)	R	MP, For 3, For 2
5)	Q	MP, For 1, For 4

On receiving this program, the machine produces the demonstration internally without printing anything. We then give the order which will cause it to print a demonstration of " [P ∧ R] ⇒ Q " in which " P ∧ R " does not appear as a hypothesis. We order the machine to "discharge" the hypothesis " P ∧ R " from the demonstration it has stored. In machine language, this command is

DT Dis, For 2

since the hypothesis " P ∧ R " appears as For 2 in the stored demonstration (" DT " stands for "Deduction Theorem"). On receiving this command the machine does the following. It calls up the first formula of the stored demonstration, namely " R ⇒ Q ". It recognizes this as a hypothesis different from the one it was asked to discharge. So it prints

---

* Notice that this is a <u>full</u> demonstration involving no use of subroutines. The reason for this will be explained a little later on.

$R \Rightarrow Q$

$[R \Rightarrow Q] \Rightarrow [[P \wedge R] \Rightarrow [R \Rightarrow Q]]$

1′) $[P \wedge R] \Rightarrow [R \Rightarrow Q]$

Here the first formula is a hypothesis (of the demonstration the machine is constructing), the second is a tautology (an instance of "$P \Rightarrow [Q \Rightarrow P]$"), and (1′) follows from the first two by Modus Ponens.

The machine then examines the second formula in the stored demonstration, namely "$P \wedge R$", which it identifies as the hypothesis to be discharged, and it prints

2′) $[P \wedge R] \Rightarrow [P \wedge R]$

which is a tautology.

Next the machine examines the third formula, "$[P \wedge R] \Rightarrow R$", which it recognizes as one of the tautologies (in fact, the only one) that it stored at the outset, so it prints

3′) $[P \wedge R] \Rightarrow [[P \wedge R] \Rightarrow R]$

which is also a tautology. The machine then examines the next formula, "R", recognizes that it follows from "$[P \wedge R] \Rightarrow R$" and "$P \wedge R$" by Modus Ponens, and prints

$[[P \wedge R] \Rightarrow [[P \wedge R] \Rightarrow R]] \Rightarrow [[[P \wedge R] \Rightarrow [P \wedge R]] \Rightarrow [[P \wedge R] \Rightarrow R]]$

$[[P \wedge R] \Rightarrow [P \wedge R]] \Rightarrow [[P \wedge R] \Rightarrow R]$

4′) $[P \wedge R] \Rightarrow R$

Here the first formula is a tautology, the second follows from the first and (3′) by Modus Ponens, and the third follows from the second and (2′) by Modus Ponens. Finally, the machine considers the last formula, "Q", recognizes that it follows from "$R \Rightarrow Q$" and "R" by Modus Ponens and prints

$[[P \wedge R] \Rightarrow [R \Rightarrow Q]] \Rightarrow [[[P \wedge R] \Rightarrow R] \Rightarrow [[P \wedge R] \Rightarrow Q]]$

$[[P \wedge R] \Rightarrow R] \Rightarrow [[P \wedge R] \Rightarrow Q]$

5′) $[P \wedge R] \Rightarrow Q$

Here the first formula is a tautology, the second follows by Modus Ponens from the first and (1′) and the third follows by Modus Ponens from the second and (4′). So, on receiving the

command " DT Dis, For 2 " our machine has printed out eleven formulas. This sequence of formulas constitutes a demonstration of " [P ∧ R] ⇒ Q " from " R ⇒ Q ".

Let us give a final example of how use of the Deduction Theorem helps generate demonstrations. Suppose we are asked to show that R ⇒ Q, ~S ⊢ ~Q ⇒ ~[S ∨ R]; i.e., to show that there exists a demonstration of " ~Q ⇒ ~[S ∨ R] " from " R ⇒ Q " and " ~S ". Because of the Deduction Theorem we can show the existence of such a demonstration by actually producing a demonstration of a different result, namely by showing that R ⇒ Q, ~S, ~Q ⊢ ~[S ∨ R]. Then our machine will print a demonstration of " ~Q ⇒ ~[S ∨ R] " from " R ⇒ Q " and " ~S " when we tell it to discharge the hypothesis, " ~Q ". The following is an outline of what will happen. We have included a program outline to explain what is going on.

	Demonstration Outline	Program Outline
1)	R ⇒ Q	Hyp
2)	~S	Hyp
3)	~Q	Dis Hyp
4)	~R	MT, For 1, For 3
5)	~S ∧ ~R	CI, For 2, For 4
6)	[~S ∧ ~R] ⇒ ~[S ∨ R]	Taut
7)	~[S ∨ R]	MP, For 6, For 5
8)	~Q ⇒ ~[S ∨ R]	DT Dis, For 3

On receiving the program corresponding to this outline, the machine first stores a full demonstration of " ~[S ∨ R] " from the hypotheses " R ⇒ Q ", " ~S " and " ~Q ". It then takes this full demonstration, considers each formula in turn (as in the preceding example) and makes the appropriate changes with the result that it prints a demonstration of " ~Q ⇒ ~[S ∨ R] " from the hypotheses " R ⇒ Q " and " ~S ".

Returning to a point we raised during our discussion of the previous example, the reason we showed you a full demonstration in that case rather than taking advantage of the subroutines that were available to us was that we wanted you to see how the machine considers

each formula of a full demonstration during the discharging process. Of course, the machine never deals with anything but full demonstrations; the subroutines were introduced purely to save us time and effort. When we give the machine a subroutine order it produces all the formulas involved, not just the last one; and if the use of a subroutine is involved in a demonstration one of whose hypotheses the machine is discharging, it will consider each of those intermediate formulas as well as the one that was the end result of the use of that subroutine. For example, in the case above, between For 3 and For 4 the machine actually produces the following two formulas:

$$[R \Rightarrow Q] \Rightarrow [\sim Q \Rightarrow \sim R]$$

$$\sim Q \Rightarrow \sim R$$

Of these, the first is a tautology and the second is obtained from the first and For 1 by Modus Ponens. During the course of the discharging process, the machine considers each of these two formulas as well as the ones that are mentioned in our demonstration outline. In this instance, the machine will print at this stage:

$$\sim Q \Rightarrow [[R \Rightarrow Q] \Rightarrow [\sim Q \Rightarrow \sim R]]$$

$$[\sim Q \Rightarrow [[R \Rightarrow Q] \Rightarrow [\sim Q \Rightarrow \sim R]]] \Rightarrow [[\sim Q \Rightarrow [R \Rightarrow Q]] \Rightarrow [\sim Q \Rightarrow [\sim Q \Rightarrow \sim R]]]$$

$$[\sim Q \Rightarrow [R \Rightarrow Q]] \Rightarrow [\sim Q \Rightarrow [\sim Q \Rightarrow \sim R]]$$

$$\sim Q \Rightarrow [\sim Q \Rightarrow \sim R]$$

In a similar way, the machine considers the two formulas that intervene between For 4 and For 5 of our outline above.

PROBLEM SET 11

In Problems 1 - 3, complete the program outline for the demonstration.

1. Show that $P \Rightarrow [Q \wedge S] \vdash P \Rightarrow [Q \vee S]$.

Demonstration Outline		Program Outline
1)	$P \Rightarrow [Q \wedge S]$	
2)	P	
3)	$Q \wedge S$	
4)	$[Q \wedge S] \Rightarrow [Q \vee S]$	
5)	$Q \vee S$	
6)	$P \Rightarrow [Q \vee S]$	

2. Show that $P \Rightarrow [Q \Leftrightarrow S],\ Q,\ R \Rightarrow [Q \Leftrightarrow S] \vdash [P \vee R] \Rightarrow S$.

Demonstration Outline		Program Outline
1)	$P \Rightarrow [Q \Leftrightarrow S]$	
2)	Q	
3)	$R \Rightarrow [Q \Leftrightarrow S]$	
4)	$P \vee R$	
5)	$[P \vee R] \Rightarrow [Q \Leftrightarrow S]$	
6)	$Q \Leftrightarrow S$	
7)	S	
8)	$[P \vee R] \Rightarrow S$	

3. Show that $\sim Q \Rightarrow [R \vee S],\ S \Leftrightarrow [P \vee R],\ \sim P \vdash \sim R \Rightarrow Q$.

Demonstration Outline		Program Outline
1)	$\sim Q \Rightarrow [R \vee S]$	
2)	$S \Leftrightarrow [P \vee R]$	
3)	$\sim P$	
4)	$\sim R$	
5)	$\sim P \wedge \sim R$	

6) $S \Rightarrow [P \vee R]$

7) $\sim[P \vee R] \Rightarrow \sim S$

8) $[\sim P \wedge \sim R] \Rightarrow \sim[P \vee R]$

9) $[\sim P \wedge \sim R] \Rightarrow \sim S$

10) $\sim S$

11) $\sim R \wedge \sim S$

12) $\sim[R \vee S] \Rightarrow Q$

13) $[\sim R \wedge \sim S] \Rightarrow \sim[R \vee S]$

14) $[\sim R \wedge \sim S] \Rightarrow Q$

15) Q

16) $\sim R \Rightarrow Q$

In Problems 4 - 9, outline a demonstration of the given result that makes use of the Deduction Theorem (even if you could have demonstrated the result without using the Deduction Theorem), and provide a program outline for that demonstration.

4. Show that $P \Rightarrow R,\ Q \Rightarrow T,\ [R \wedge T] \Leftrightarrow S \vdash [P \wedge Q] \Rightarrow S$.

5. Show that $S \Rightarrow P,\ Q \Rightarrow R,\ S \vdash [P \Rightarrow Q] \Rightarrow R$.

6. Show that $R \Rightarrow T,\ \sim T \Leftrightarrow S,\ [R \wedge \sim S] \Rightarrow \sim Q \vdash R \Rightarrow \sim Q$.

7. Show that $P \Rightarrow R,\ [P \wedge R] \Rightarrow S,\ \sim S \vee Q \vdash P \Rightarrow Q$.

8. Show that $P \Rightarrow Q,\ P \Rightarrow [Q \Rightarrow R],\ Q \Rightarrow [R \Rightarrow S] \vdash P \Rightarrow S$.

9. Show that $[R \wedge \sim Q] \Rightarrow P,\ [T \Rightarrow S] \Leftrightarrow [R \Rightarrow Q],\ R \vdash [\sim P \vee [T \Rightarrow S]] \Rightarrow Q$.

10. For each of the following claims decide whether or not it is a logical consequence of the assumptions. Do this by identifying well formed formulas which are patterns for those sentences, outlining an appropriate demonstration (accompanied by a program outline) when it exists, and showing that a certain formula is <u>not</u> a tautology when no demonstration exists.

Assumptions:

1) If Jim's arm hurts, he pitches badly.

2) If Jim pitches badly, the team loses.

3) If Jim pitches badly or the team loses, the crowd does not applaud.

Claims:

a) If the crowd applauds, then Jim's arm does not hurt.

b) If the team loses, Jim's arm hurts.

c) If the crowd does not applaud, then the team loses.

d) If Jim pitches badly, then his arm hurts and the crowd applauds.

e) If Jim's arm does not hurt and he pitches badly and the crowd applauds, then the team loses.

- • • -

2.8 Using the Deduction Theorem

So far all the examples of the use of the Deduction Theorem that you have seen have been rather simple ones in the sense that all the demonstrations have ended as soon as the discharge command has been executed. As it happens, there are more complicated situations in which the Deduction Theorem is used at an intermediate stage of a demonstration. Let us illustrate with an example.

Example 1: Show that $\sim P \vee R,\ [P \wedge R] \Rightarrow Q,\ Q \Rightarrow P,\ [P \Leftrightarrow Q] \Rightarrow \sim Q \vdash \sim P \vee \sim R$.

At first glance it may not be clear where, or how, to begin. We need to do a little advance planning. What is it we are trying to prove? "$\sim P \vee \sim R$". Nothing like that appears to the left of the turnstile. Let's see if there's anything to be gained by considering formulas that are equivalent to "$\sim P \vee \sim R$". The first such equivalent formula you will think of will probably be "$\sim[P \wedge R]$". Now we're getting somewhere. We could infer "$\sim[P \wedge R]$" from the hypothesis "$[P \wedge R] \Rightarrow Q$" by Modus Tollens if we had "$\sim Q$"; and we could infer "$\sim Q$" by Modus Ponens from the hypothesis "$[P \Leftrightarrow Q] \Rightarrow \sim Q$" if we had "$P \Leftrightarrow Q$". The "$Q \Rightarrow P$" part of "$P \Leftrightarrow Q$" is a hypothesis, so the only thing we are lacking is "$P \Rightarrow Q$". At this point we recall that "$\sim P \vee R$" is equivalent to "$P \Rightarrow R$", and then it's not difficult to see how we could use the Deduction Theorem to show that

$$P \Rightarrow R,\ [P \wedge R] \Rightarrow Q \vdash P \Rightarrow Q.$$

So our plan of attack is to reverse the above line of reasoning. Here's the demonstration outline.

	Demonstration Outline	Program Outline
1)	$\sim P \vee R$	Hyp
2)	$[P \wedge R] \Rightarrow Q$	Hyp
3)	P	Dis Hyp
4)	$[\sim P \vee R] \Leftrightarrow [P \Rightarrow R]$	Taut
5)	$P \Rightarrow R$	GSP, For 4, For 1 ✶

✶ See footnote on page 86. Either MPB or GSP would be correct here. In the text we will use GSP.

6)	R	MP, For 5, For 3
7)	$P \wedge R$	CI, For 3, For 6
8)	Q	MP, For 2, For 7
9)	$P \Rightarrow Q$	DT Dis, For 3
10)	$Q \Rightarrow P$	Hyp
11)	$P \Leftrightarrow Q$	CI, For 9, For 10
12)	$[P \Leftrightarrow Q] \Rightarrow \sim Q$	Hyp
13)	$\sim Q$	MP, For 12, For 11
14)	$\sim[P \wedge R]$	MT, For 2, For 13
15)	$\sim[P \wedge R] \Leftrightarrow [\sim P \vee \sim R]$	Taut
16)	$\sim P \vee \sim R$	GSP, For 15, For 14

This outline looks fine, but there is the potential for a problem here. To understand what that problem might be, let us review what the machine does when it is given a program such as the one that is outlined above. As soon as it receives a "Dis Hyp" input formula it knows that it has to store the results of its work until such time as a discharge command is given. When it receives this command, as for example in step (9) of the foregoing outline, it takes the whole of the demonstration it has stored up to that point, and for each formula in this demonstration it forms a conditional. (In this particular case, all these conditionals will have For 3 as antecedent.) The machine then inserts formulas in between these conditionals according to rules that were discussed in the last section, and before proceeding to the execution of the next command (in this case step (10)) it prints out this entire sequence of formulas.

So at step (9) the machine will print a demonstration of "$P \Rightarrow Q$" from the hypotheses "$\sim P \vee R$" and "$[P \wedge R] \Rightarrow Q$". Our potential problem is raised by what we did at step (14). There we called for For 2, "$[P \wedge R] \Rightarrow Q$". Now, since this is a hypothesis of the demonstration the machine printed at step (9), we know it appears somewhere in that demonstration, but we might not be so lucky in other demonstrations. We could easily find ourselves in a situation where we are calling for a formula which is not a hypothesis and which we

have listed before the execution of a discharge order. How do we know that such a formula appears in the demonstration the machine prints out? How do we know that there are formulas in that printout corresponding to all the formulas we have in our outline?

There are two ways we could try to deal with this problem. The first is to take the view that there's no way we can know how the machine printout at step (9) corresponds to our outline and so the only formula we know for sure is there is the last one, " P ⇒ Q ". Consequently, at step (14) when we need For 2 we must reprogram step (2) in order to restore that formula. But this is such a nuisance. Why should we do all this work when we have a machine that could do it for us. Let us modify the machine in the following way.

The source of all this trouble is the discharged hypothesis. What we need to do is avoid that hypothesis and any formula derived from it. So when we give the machine a program which involves a "Dis Hyp" input formula, let's make it so that the machine forms and stores the demonstration as before while it awaits a discharge order. On receipt of such an order, the machine will first print all formulas in the stored demonstration except for the hypothesis to be discharged and any formula derived from that hypothesis. All the formulas printed in this way will still be identified by the formula numbers of our original outline. For example, in our example above, at step (9) the machine will begin by printing formulas (1), (2), and (4) and continue to identify them by these numbers. None of the other formulas will be printed because they are either the hypothesis being discharged (For 3) or their derivation depends on it (For 5, For 6, For 7, For 8). Then, having printed For 4, the machine will go through its hypothesis-discharging procedure. That is, it will take the entire stored demonstration, make the appropriate conditionals, insert the appropriate formulas, and continue printing after For 4 by printing a demonstration of " P ⇒ Q ". For our convenience, the machine will identify this formula as For 9. With this modification, the machine will experience no difficulty in executing commands that follow a discharge order, provided we call neither for the discharged hypothesis nor for any formula derived from it. In our particular example, we would not be allowed to call for formulas (3) and (5)-(8).

All this may seem very complicated to you. In particular, you would be justified in asking how you are supposed to remember which formulas are still available after a discharge order. One way of playing it safe is as follows: Since no trouble can occur before the hypothesis to be discharged, you may always use a formula that precedes that hypothesis. The formulas from that point until the execution of the discharge order may be usable at a later stage in the demonstration, or then again they may not be. To be safe, do not make later use of any formula from the hypothesis that is discharged to the last formula resulting from the execution of the discharge order, including the former but excluding the latter. If it happens that you have cause to call for one of these "risky" formulas, if it does not depend on a discharged hypothesis for its derivation you will always be justified in reintroducing it at the time it is needed. To help you play it safe in this way we will henceforth make use of a little aide-mémoire (i.e., an aid for your memory) to indicate which formulas are risky.

Suppose that at step (m+1) in a demonstration outline we order the discharge of For n, then to the right of the Program Outline column in step (m+1) we will write "(n)-(m)" as a reminder that formulas (n)-(m) are risky and should be avoided. So, for example, in Example 1 above, at step (9) we would have the following:

Demonstration Outline		Program Outline	
⋮		⋮	
9)	$P \Rightarrow Q$	DT Dis, For 3	(3)-(8)

Thus, "(3)-(8)" is strictly a precautionary reminder to ourselves; it has nothing to do with the instructions we give the machine. It reminds us that risks are involved if we make later use of any of formulas (3) - (8), and we might be better off avoiding them entirely.

We now provide a few more examples of the intermediate use of the Deduction Theorem during the course of a longer demonstration. Examination of these examples will familiarize you with the various situations that can arise in such circumstances.

Example 2: Show that $P \vee R,\ [Q \wedge R] \Rightarrow S,\ \sim S \wedge R \vdash \sim Q$.

To illustrate our point we provide a demonstration that is needlessly long. In fact, you will show in Problem 17 of Problem Set 12 that the first hypothesis is unnecessary.

	Demonstration Outline	Program Outline	
1)	$Q \wedge P$	Dis Hyp	
2)	Q	RCS, For 1	
3)	R	Dis Hyp	
4)	$Q \wedge R$	CI, For 2, For 3	
5)	$[Q \wedge R] \Rightarrow S$	Hyp	
6)	S	MP, For 5, For 4	
7)	$R \Rightarrow S$	DT Dis, For 3	(3)-(6)
8)	$[Q \wedge P] \Rightarrow [R \Rightarrow S]$	DT Dis, For 1	(1)-(7)
9)	$\sim S \wedge R$	Hyp	
10)	$[\sim S \wedge R] \Leftrightarrow \sim [R \Rightarrow S]$	Taut	
11)	$\sim [R \Rightarrow S]$	GSP, For 10, For 9	
12)	$\sim [Q \wedge P]$	MT, For 8, For 11	
13)	$\sim [Q \wedge P] \Leftrightarrow [P \Rightarrow \sim Q]$	Taut	
14)	$P \Rightarrow \sim Q$	GSP, For 13, For 12	
15)	$\sim S$	RCS, For 9	
16)	$[Q \wedge R] \Rightarrow S$	Hyp	
17)	$\sim [Q \wedge R]$	MT, For 16, For 15	
18)	$\sim [Q \wedge R] \Leftrightarrow [R \Rightarrow \sim Q]$	Taut	
19)	$R \Rightarrow \sim Q$	GSP, For 18, For 17	
20)	$[P \vee R] \Rightarrow \sim Q$	IC, For 14, For 19	
21)	$P \vee R$	Hyp	
22)	$\sim Q$	MP, For 20, For 21	

Notice that formula (5) is identical with formula (16). This occurred because formula (5) was needed in order to infer formula (17). However, formula (5) is on our list of risky

formulas, as we see from our aide-mémoire at step (7). Hence, we reintroduced it as formula (16), the justification being that it was a hypothesis of the main demonstration. (Of course, if we had thought ahead and seen this difficulty before it arose we could have written For 5 as For 1, and then it would never have been caught up in a use of the Deduction Theorem. Indeed, in general, it is often a good idea when constructing a demonstration to list all the hypotheses that are <u>not</u> going to be discharged before listing the first of those that are.)

A quick glance down the Program Outline column shows that, since we have used none of formulas (1) - (7), we have not used any risky formula. After step (7), formulas (3) - (6) are risky but the use of formula (1) at step (8) is O.K. After step (8), all the formulas (1) - (7) are on the risky list, and we see that from step (9) on none of these formulas is referred to.

Example 3: Show that $S,\ [S \wedge P] \Rightarrow Q,\ [[\tilde{}P \vee Q] \wedge R] \Rightarrow Q \vdash [\tilde{}P \wedge \tilde{}R] \vee Q$.

	Demonstration Outline	Program Outline	
1)	S	Hyp	
2)	P	Dis Hyp	
3)	$S \wedge P$	CI, For 1, For 2	
4)	$[S \wedge P] \Rightarrow Q$	Hyp	
5)	Q	MP, For 4, For 3	
6)	$P \Rightarrow Q$	DT Dis, For 2	(2) - (5)
7)	$[P \Rightarrow Q] \Leftrightarrow [\tilde{}P \vee Q]$	Taut	
8)	$\tilde{}P \vee Q$	GSP, For 7, For 6	
9)	R	Dis Hyp	
10)	$[\tilde{}P \vee Q] \wedge R$	CI, For 8, For 9	
11)	$[[\tilde{}P \vee Q] \wedge R] \Rightarrow Q$	Hyp	
12)	Q	MP, For 11, For 10	
13)	$R \Rightarrow Q$	DT Dis, For 9	(9) - (12)

14)	$[P \vee R] \Rightarrow Q$	IC, For 6, For 13
15)	$[[P \vee R] \Rightarrow Q] \Leftrightarrow [[\sim P \wedge \sim R] \vee Q]$	Taut
16)	$[\sim P \wedge \sim R] \vee Q$	GSP, For 15, For 14

Check through the Program Outline and make sure that at no stage have any formulas which are risky at that stage been used.

Example 4: As a final example in this section, we outline a "demonstration" which shows fallaciously (i.e., by an argument that contains a logical mistake) that $[S \wedge P] \Rightarrow Q,\ Q \Rightarrow P \vdash S \Rightarrow Q$.

	Demonstration Outline	Program Outline	
1)	S	Dis Hyp	
2)	P	Dis Hyp	
3)	$S \wedge P$	CI, For 1, For 2	
4)	$[S \wedge P] \Rightarrow Q$	Hyp	
5)	Q	MP, For 4, For 3	
6)	$P \Rightarrow Q$	DT Dis, For 2	(2)-(5)
7)	$Q \Rightarrow P$	Hyp	
8)	P	MP, For 7, For 5	
9)	$S \Rightarrow P$	DT Dis, For 1	(1)-(8)
10)	$S \Rightarrow Q$	SI, For 9, For 6	

The fallacy (i.e., the mistake) lies partially in the fact that at step (8) use was made of formula (5). But by the time we arrive at step (8) formula (5) has appeared on our list of risky formulas. Indeed, calling on formula (5) at step (8) is illegitimate. You will show in the following exercise that this mistake is irremediable (i.e., there is no demonstration of "Q" from "$[S \wedge P] \Rightarrow Q$" and "$Q \Rightarrow P$").

Exercise 9: (i) Show that "$[[S \wedge P] \Rightarrow Q] \Rightarrow [[Q \Rightarrow P] \Rightarrow [S \Rightarrow Q]]$" is <u>not</u> a tautology.

(ii) By using the Deduction Theorem, explain why part (i) shows that "$[S \wedge P] \Rightarrow Q,\ Q \Rightarrow P \vdash S \Rightarrow Q$" is not true.

In the following problem set, and indeed in all your later work using the propositional calculus, if a demonstration includes a use of the Deduction Theorem, it is your responsibility to insure that every formula used after a discharge order is in fact available for use. The safest way to insure this is to avoid all the risky ones.

PROBLEM SET 12

In Problems 1 - 3, fill in the missing formulas and program steps.

1. Show that $\sim P \Rightarrow [Q \wedge R],\ P \Rightarrow \sim S \vdash S \Rightarrow Q$.

	Demonstration Outline	Program Outline
1)	$\sim P \Rightarrow [Q \wedge R]$	
2)	$P \Rightarrow \sim S$	
3)		Dis Hyp
4)		CPI, For 2
5)	$\sim P$	
6)		MP, For 1, For 5
7)	Q	
8)	$S \Rightarrow Q$	

2. Show that $P \Rightarrow Q,\ [\sim P \vee R] \Rightarrow S \vdash Q \vee S$.

	Demonstration Outline	Program Outline
1)	$P \Rightarrow Q$	
2)		Hyp
3)	$\sim S$	
4)		MT, For 2, For 3
5)		Taut
6)	$P \wedge \sim R$	
7)		RCS, For 6
8)		MP, For 1, For 7

9)		DT Dis, For 3	(3) - (8)
10)	$[\sim S \Rightarrow Q] \Leftrightarrow [Q \vee S]$		
11)	$Q \vee S$		

3. Show that $\sim P \Rightarrow Q,\ Q \Rightarrow [R \wedge \sim S] \vdash [S \vee \sim Q] \Rightarrow P$.

	Demonstration Outline	Program Outline
1)	$\sim P \Rightarrow Q$	
2)		Hyp
3)	S	
4)		Taut
5)	$\sim R \vee S$	
6)		Taut
7)		GSP, For 6, For 5
8)	$\sim Q$	
9)	P	
10)	$S \Rightarrow P$	
11)	$\sim Q \Rightarrow P$	
12)	$[S \vee \sim Q] \Rightarrow P$	

In Problems 4 - 6, outline a demonstration of the given result that makes use of the Deduction Theorem and provide a program outline for that demonstration.

4. Show that $\sim R \Rightarrow P,\ [P \wedge \sim S] \Rightarrow Q \vdash \sim [R \vee S] \Rightarrow Q$.

5. Show that $P \Rightarrow [Q \Rightarrow R] \vdash Q \Rightarrow [P \Rightarrow R]$.

6. Show that $Q \Rightarrow R \vdash [\sim Q \Rightarrow \sim P] \Rightarrow [P \Rightarrow R]$.

Review: In Problems 7 - 16, outline a demonstration of the given result and provide a program outline for that demonstration.

7. Show that $P \Rightarrow Q,\ Q \Rightarrow \sim R,\ R,\ S \Rightarrow P \vdash \sim S$.

8. Show that $S \Rightarrow [P \wedge \sim Q],\ \sim S \Rightarrow \sim R,\ R \vdash \sim Q \wedge P$.

9. Show that $P \Rightarrow Q,\ Q \Rightarrow R,\ [P \Rightarrow R] \Rightarrow \sim S,\ \sim S \Rightarrow T \vdash T$.

10. Show that $T \Rightarrow Q,\ P \Rightarrow Q,\ R \Rightarrow Q \vdash [[P \vee R] \vee T] \Rightarrow Q$.

11. Show that $R \Rightarrow S,\ [R \Rightarrow \sim P] \Rightarrow Q,\ S \Rightarrow \sim P,\ T \vdash Q \wedge T$.

12. Show that $P \Rightarrow Q,\ R \Rightarrow S,\ P \vee R,\ \sim Q \vdash S$.

13. Show that $P \vee Q,\ \sim R \Rightarrow \sim S,\ [Q \vee P] \Rightarrow S,\ \sim[T \wedge U] \Rightarrow \sim R \vdash U \wedge T$.

14. Show that $\sim P \Rightarrow \sim R,\ \sim S,\ P \Rightarrow S,\ \sim R \Rightarrow Q \vdash Q$.

15. Show that $\sim P,\ \sim Q,\ R \Rightarrow P \vdash \sim[Q \vee R]$.

16. Show that $\sim[T \wedge R],\ \sim T \Rightarrow S,\ \sim R \Rightarrow S \vdash S$.

17. Show that $[Q \wedge R] \Rightarrow S,\ \sim S \wedge R \vdash \sim Q$.

- o O o -

2.9 Object Language and Metalanguage*; The Principle of Indirect Inference

Suppose you are a writer and you have been asked to write a book about the French language for an English-speaking audience. You will be writing about French, and you will be doing it in English. You will express yourself in the English language as you describe how the French language is organized, how its verbs are conjugated, how its adjectives are inflected, etc. You will probably also present examples of French, such as

(1) La plume de ma tante est perdue.

The object of discourse in (1) is my aunt's pen, and what is stated is that it is lost. However, you, as the writer of the book, might want to provide a discussion in English about this French sentence. You might wish to talk about the structure of the sentence, pointing out how the verb is used, or the agreement in gender of the final adjective with the subject of the sentence. That is, from your point of view as the writer, the object of discourse is not my aunt's pen, but rather the sentence (1) itself.

In a case such as this, we call French the object language, and the language (in this case, English) in which we discuss the object language, the metalanguage. Even if one were to write a book in English about the English language, it would still be necessary

* The Greek prefix "meta" means "above" or "beyond".

to bear in mind the object language—metalanguage relation. Indeed, one would have to use some device for indicating when a sentence is regarded as being in the object language and when not. A commonly accepted device is to enclose in quotation marks words or phrases about which one is talking. The footnote on page 125 provides an example of this.

We may draw an analogy between the problems of writing a book in English about the French language and those of writing this book in English about the propositional calculus. In our case, the language of the propositional calculus is the <u>object language</u>, and the <u>metalanguage</u> is English, augmented by a few convenient symbols such as "$\vdash$", "$\mathcal{S}$", etc.

The basic vocabulary of the propositional calculus consists of

a) Propositional variables: P, Q, R, ...

b) Connectives: $\sim, \wedge, \vee, \Rightarrow$.

c) Grouping symbols: [,].

The sentences of this language are simply well formed formulas, which were first described in Chapter 1. Then it is clear that we may consider a demonstration to be a certain type of paragraph in the object language. The following is a statement <u>about</u> certain sorts of wffs, that is, it is a statement about the object language and as such it is part of the metalanguage.

(*) A wff is a tautology iff it has a hypothesis-free demonstration.

A convenient way of abbreviating statement (*) is as follows:

(**) A wff $\mathcal{S}$ is a tautology iff $\vdash \mathcal{S}$.

Although we have used two new symbols (namely "$\mathcal{S}$" and "$\vdash$"), (**) is still in the metalanguage. Observe that neither "$\mathcal{S}$" nor "$\vdash$" is listed above as being part of the basic vocabulary of the object language. These symbols are metasymbols that we add to English for their convenience in helping us talk about the propositional calculus. For this reason, instead of writing

(◇) $P, P \Rightarrow Q \vdash Q$,

strictly speaking we should write

(◇◇) "P", "$P \Rightarrow Q$" $\vdash$ "Q".

since what we mean by (◇) is that there is a demonstration of "Q" from "P" and "$P \Rightarrow Q$", which is a statement in the metalanguage. If you mention something from the object language (e.g., "P") during the course of a statement in the metalanguage you really ought to enclose it in quotation marks. However, in this case, since it is difficult to see how any confusion could arise, we will continue to be lax (and brief!) and prefer (◇) to (◇◇). Notice that statement (**) is just a restatement of the Principle of Hypothesis-Free Demonstration (page 100). This principle and the other two we have met thus far (the Principle of Unnecessary Hypotheses, page 61, and the Deduction Theorem, page 101) are all stated in the metalanguage and are statements <u>about</u> something in the object language. For example, the Deduction Theorem tells us something about demonstrations, that is, about certain sequences of "sentences" in the object language. Here is another such result, known as the <u>Principle of Indirect Inference</u>. Here $G_1, \ldots, G_n$, $\mathcal{B}$, and C are wffs and we allow the possibility of there being no G's.

The Principle of Indirect Inference: $G_1, \ldots, G_n, \sim\mathcal{B} \vdash C \wedge \sim C$ iff $G_1, \ldots, G_n \vdash \mathcal{B}$.

This principle is easily justified using the Deduction Theorem. As usual in situations involving "iff" we tackle this task in two parts:

i) First we must show: If $G_1, \ldots, G_n, \sim\mathcal{B} \vdash C \wedge \sim C$, then $G_1, \ldots, G_n \vdash \mathcal{B}$.

If $G_1, \ldots, G_n, \sim\mathcal{B} \vdash C \wedge \sim C$, then by the Deduction Theorem, $G_1, \ldots, G_n \vdash \sim\mathcal{B} \Rightarrow [C \wedge \sim C]$. That is, there is a demonstration whose hypotheses are $G_1, \ldots, G_n$ and whose concluding formula is

r) $\sim\mathcal{B} \Rightarrow [C \wedge \sim C]$

Imagine such a demonstration before you. Continue as follows:

r+1)	$\sim[C \wedge \sim C] \Rightarrow \mathcal{B}$	CPI, For r
r+2)	$\sim[C \wedge \sim C]$	Taut
r+3)	$\mathcal{B}$	MP, For(r+1), For(r+2)

and we have a demonstration of $\mathcal{B}$ from the hypotheses $G_1, \ldots, G_n$.

ii) Next we must show that: If $G_1, \ldots, G_n \vdash \mathcal{B}$, then $G_1, \ldots, G_n, \sim\mathcal{B} \vdash C \wedge \sim C$.

If $G_1, \ldots, G_n \vdash \mathcal{B}$, then there is a demonstration whose hypotheses are $G_1, \ldots, G_n$ and whose concluding formula is

m) $\mathcal{B}$

Imagine such a demonstration before you. Continue as follows:

m+1)	$\sim\mathcal{B}$	Hyp
m+2)	$\mathcal{B} \wedge \sim\mathcal{B}$	CI, For m, For(m+1)
m+3)	$[\mathcal{B} \wedge \sim\mathcal{B}] \Rightarrow [C \wedge \sim C]$	Taut
m+4)	$C \wedge \sim C$	MP, For(m+3), For(m+2)

and we have a demonstration of $C \wedge \sim C$ from the hypotheses $G_1, \ldots, G_n, \mathcal{B}$.

Arguments that appeal to the Principle of Indirect Inference are sometimes called "contradiction arguments", because the formula

(1) $C \wedge \sim C$

is called a <u>contradiction</u>. A contradiction is a formula whose negation is a tautology, and and so all contradictions are equivalent to each other in the same way that all tautologies are equivalent to each other. Hence there is nothing special about the contradiction (1). An examination of our justification of the Principle of Indirect Inference above should make it clear that if (1) is replaced by any other contradiction the argument is still sound. That is, if K is any contradiction, we can show that

$G_1, \ldots, G_n, \sim\mathcal{B} \vdash K$ iff $G_1, \ldots, G_n \vdash \mathcal{B}$.

Arguments that use the Principle of Indirect Inference have certain similarities with those that use the Deduction Theorem. Recall that in order to show that $G_1, \ldots, G_n \vdash \mathcal{B} \Rightarrow C$ the Deduction Theorem allows us to deal with the easier situation where we have an extra hypothesis. If we can show the existence of a demonstration of C from the hypotheses $G_1, \ldots, G_n, \mathcal{B}$, then the Deduction Theorem enables us to infer the existence of a demonstration of $\mathcal{B} \Rightarrow C$ from the hypotheses $G_1, \ldots, G_n$. Indeed, the proof of the Deduction Theorem tells us exactly how to obtain the second demonstration from the first. In fact it

is the justification of the Deduction Theorem (not the theorem itself) that enabled us to construct a Deduction Theorem subroutine.

Similarly, the Principle of Indirect Inference assures us that a demonstration of $\mathcal{B}$ from the hypotheses $G_1, \ldots, G_n$ exists iff a demonstration of a contradiction $C \wedge \tilde{}C$ from the hypotheses $G_1, \ldots, G_n, \tilde{}\mathcal{B}$ exists. And the justification of the principle tells us how to construct one demonstration from the other.

Let us now give some examples of how the Principle of Indirect Inference may be used in the construction of demonstrations. These examples will also show how, with the addition of one more command, our machine can be modified so that it does most of the work for us.

Example 1: Show that $Q \Rightarrow \tilde{}P,\ \tilde{}R \Rightarrow [P \wedge Q] \vdash R$.

We present the outline of a demonstration which makes use of the Principle of Indirect Inference, and we will explain the program outline afterwards.

	Demonstration Outline	Program Outline
1)	$Q \Rightarrow \tilde{}P$	Hyp
2)	$\tilde{}R \Rightarrow [P \wedge Q]$	Hyp
3)	$\tilde{}R$	Dis Hyp
4)	$P \wedge Q$	MP, For 2, For 3
5)	Q	LCS, For 4
6)	$\tilde{}P$	MP, For 1, For 5
7)	P	RCS, For 4
8)	$P \wedge \tilde{}P$	CI, For 7, For 6
9)	R	Contra Dis, For 3 (3)-(8)

Now it should be clear what we design our machine to do upon receiving the order "Contra Dis, For 3" (which is machine language telling the machine to use the Principle of Indirect Inference and discharge formula (3)—"Contra" is short for "Contradiction"). Looking back at our justification of the principle, we see that we should design the machine's response so that, first of all, it will go through a normal Deduction Theorem "discharge procedure"

and print a demonstration of "$\sim R \Rightarrow [P \wedge \sim P]$" from "$Q \Rightarrow \sim P$" and "$\sim R \Rightarrow [P \wedge Q]$" on the basis of formulas (1) - (8). This printout will terminate with the formula

8′) $\sim R \Rightarrow [P \wedge \sim P]$.

Then, making use of the Contrapositive Inference subroutine, the machine will print a sequence of formulas terminating with

8″) $\sim [P \wedge \sim P] \Rightarrow R$.

Next, it will print the negation of formula (8), i.e., of the formula it printed immediately prior to starting execution of the Indirect Inference procedure, namely

8‴) $\sim [P \wedge \sim P]$.

We know that this formula is a tautology because it was our recognition of the fact that "$P \wedge \sim P$" was a contradiction that prompted us to give the "Contradiction, discharge" order in the first place. Hence the machine is justified in printing formula (8‴). Then, using Modus Ponens on formulas (8″) and (8‴), the machine prints

8ᴵᵛ) R.

Notice that, since use of the Principle of Indirect Inference involves use of the Deduction Theorem, some formulas of the demonstration will be at risk if we are considering any later use of them. Just as we did in the case of the Deduction Theorem, we remind ourselves which formulas these are by the notation "(3) - (8)" at step (9).

Example 2: Show indirectly (i.e., by calling on the Principle of Indirect Inference) that $P \Rightarrow \sim S,\ S \vee \sim R,\ \sim [Q \vee \sim R] \vdash \sim P$.

	Demonstration Outline	Program Outline
1)	P	Dis Hyp
2)	$P \Rightarrow \sim S$	Hyp
3)	$\sim S$	MP, For 2, For 1
4)	$S \vee \sim R$	Hyp
5)	$\sim S \wedge [S \vee \sim R]$	CI, For 3, For 4
6)	$[\sim S \wedge [S \vee \sim R]] \Rightarrow \sim R$	Taut

7)	$\sim R$	MP, For 6, For 5	
8)	$\sim[Q \vee \sim R]$	Hyp	
9)	$\sim[Q \vee \sim R] \Leftrightarrow [\sim Q \wedge R]$	Taut	
10)	$\sim Q \wedge R$	GSP, For 9, For 8	
11)	R	LCS, For 10	
12)	$R \wedge \sim R$	CI, For 11, For 7	
13)	$\sim P$	Contra Dis, For 1	(1) - (12)

Indirect Inference is a very powerful tool in the demonstration builder's armory. It is always worth bearing in mind, and if you cannot see how to proceed when asked to outline a demonstration of a given result, it is often most profitable to assume the negation of what you are trying to prove and see if you can arrive at a contradiction. This type of argumentation has a very long history, and we have examples of its use by the Ancient Greeks. In classical logic it was known as "reductio ad absurdum", which may be translated as "reduction to an absurdity", the absurdity, of course, being the contradiction. In other words, if one wants to show that some formula $\mathcal{S}$ is a logical consequence of a set of hypotheses $G_1, \ldots, G_n$, it suffices to assume the negation of $\mathcal{S}$ (i.e., $\sim\mathcal{S}$) and show that this leads to an absurdity.

PROBLEM SET 13

1. Provide a justification of the following result:

Principle of Contradictory Hypotheses: If $\mathcal{C}$ and $\mathcal{U}$ are wffs, then $\mathcal{C} \wedge \sim\mathcal{C} \vdash \mathcal{U}$.

Stated in words, from a contradiction <u>any</u> well formed formula may be inferred.

In Problems 2 - 5, complete the program outline of the demonstration.

2. Show indirectly that $R \vee [P \wedge Q], \sim Q \vdash R$.

	Demonstration Outline	Program Outline
1)	$R \vee [P \wedge Q]$	
2)	$\sim R$	
3)	$\sim R \wedge [R \vee [P \wedge Q]]$	
4)	$[\sim R \wedge [R \vee [P \wedge Q]]] \Rightarrow [P \wedge Q]$	
5)	$P \wedge Q$	
6)	Q	
7)	$\sim Q$	
8)	$Q \wedge \sim Q$	
9)	R	

3. Show indirectly that $P \Rightarrow S, R \Rightarrow \sim S, R \vdash \sim P$.

	Demonstration	Program Outline
1)	P	
2)	$P \Rightarrow S$	
3)	S	
4)	R	
5)	$R \Rightarrow \sim S$	
6)	$\sim S$	
7)	$S \wedge \sim S$	
8)	$\sim P$	

4. Show indirectly that $S \wedge \sim T,\ S \Rightarrow \sim R,\ \sim P \Rightarrow Q \vdash R \Rightarrow Q$.

Demonstration Outline	Program Outline
1) $\sim[R \Rightarrow Q]$	
2) $\sim[R \Rightarrow Q] \Leftrightarrow [R \wedge \sim Q]$	
3) $R \wedge \sim Q$	
4) R	
5) $S \wedge \sim T$	
6) S	
7) $S \Rightarrow \sim R$	
8) $\sim R$	
9) $R \wedge \sim R$	
10) $R \Rightarrow Q$	

5. Show indirectly that $R \Rightarrow \sim S,\ Q \Rightarrow R \vdash S \Rightarrow \sim Q$.

Demonstration Outline	Program Outline
1) $\sim[S \Rightarrow \sim Q]$	
2) $\sim[S \Rightarrow \sim Q] \Leftrightarrow [S \wedge Q]$	
3) $S \wedge Q$	
4) S	
5) Q	
6) $Q \Rightarrow R$	
7) R	
8) $R \Rightarrow \sim S$	
9) $\sim S$	
10) $S \wedge \sim S$	
11) $S \Rightarrow \sim Q$	

In Problems 6 - 11, outline a demonstration which makes use of the Principle of Indirect Inference, and provide a program outline for that demonstration.

6. Show that $P \Rightarrow [Q \wedge R]$, $P \wedge Q \vdash S \Rightarrow R$.

7. Show that $P \Rightarrow \sim Q$, $\sim R \Rightarrow Q \vdash P \Rightarrow R$.

8. Show that $\sim P \Rightarrow Q$, $T \Rightarrow \sim P$, $\sim [Q \vee R] \vdash \sim T$.

9. Show that $R \Rightarrow \sim Q$, $[\sim P \Rightarrow S] \Rightarrow R$, $\sim Q \Rightarrow \sim S$, $\sim P \vdash \sim [\sim P \Rightarrow S]$.

10. Show that $R \Rightarrow \sim Q$, $\sim [S \wedge \sim R] \vdash Q \Rightarrow \sim S$.

11. Show that $\sim P \Rightarrow Q$, $Q \Rightarrow [R \Rightarrow S]$, $\sim S \vdash R \Rightarrow P$.

In Problems 12 - 16, outline a demonstration using any method you wish, and provide a program outline for that demonstration.

12. Show that $Q \Rightarrow P$, $\sim T \Rightarrow S$, $\sim Q \Rightarrow \sim S \vdash \sim [\sim P \Rightarrow R] \Rightarrow T$.

13. Show that $P \Rightarrow \sim Q$, $R \Rightarrow \sim P$, $Q \vee R \vdash \sim P$.

14. Show that $\sim P \Rightarrow \sim S$, $P \Rightarrow R$, $R \Rightarrow \sim T \vdash S \Rightarrow \sim T$.

15. Show that $\sim P \Rightarrow T$, $\sim T$, $P \Rightarrow [Q \wedge S] \vdash S \wedge Q$.

16. Show that $R \vee S$, $\sim P$, $Q \vee \sim R$, $P \Leftrightarrow Q \vdash S$.

- o O o -

See Appendix III for material that will help you review for the test on the second part of Chapter 2.

Aftermath...

> We are starting a cat ranch in Lacon with 100,000 cats. Each cat will average 12 kittens a year. The catskins will sell for 30 cents each. One hundred men can skin 5000 cats a day. We figure a daily profit of over $10,000. Now what shall we feed the cats? We will start a rat ranch next door with one million rats. The rats will breed 12 times faster than the cats. So we will have 4 rats to feed each day to each cat. Now, what shall we feed the rats? We will feed the rats the carcasses of the cats after they have been skinned. NOW GET THIS! We feed the rats to the cats and the cats to the rats and get the skins for nothing.
>
> —An 1875 newspaper advertisement, quoted in GEOGRAPHICAL REVIEW.

Chapter 3

THE PROPOSITIONAL CALCULUS

3.1 A Review

By now you have realized that the foregoing conversation about machines that print demonstrations was simply a way to introduce the ideas of the propositional calculus and induce you to think carefully about them. Along the way, we have had to think about such things as the special language of the propositional calculus, about what a well formed formula is, what a rule of inference is and how it is used, about what demonstrations are and how they are made. Let us review all of this by stating the essential points of the development up through the introduction of demonstrations. Where appropriate, we have made these statements more concisely than we did when the ideas were first introduced.

The propositional calculus consists of the following:

1) A language whose symbols include:

   a) propositional variables P, Q, R, ...

   b) logical connectives $\sim$, $\wedge$, $\vee$, $\Rightarrow$

   c) auxiliary symbols], [

2) Rules for well formed formulas:

   a) A propositional variable standing alone is a well formed formula. Well formed formulas of this type are the atomic formulas.

   b) If $\mathcal{S}$ is a well formed formula, then so is $\sim\mathcal{S}$.

   c) If $\mathcal{S}$ and $\mathcal{T}$ are well formed formulas, then so are $[\mathcal{S} \wedge \mathcal{T}]$, $[\mathcal{S} \vee \mathcal{T}]$, and $[\mathcal{S} \Rightarrow \mathcal{T}]$.

3) A rule of inference, called Modus Ponens:

   If $\mathcal{S}$ and $\mathcal{T}$ are well formed formulas, then from $[\mathcal{S} \Rightarrow \mathcal{T}]$ and $\mathcal{S}$ we may infer $\mathcal{T}$.

Our definition of "demonstration" said essentially the following:

A sequence of well formed formulas $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ is a demonstration of $\mathcal{S}$ from the hypotheses $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$ provided $\mathcal{S}_k$ is $\mathcal{S}$ and, for each i, $\mathcal{S}_i$ is either a tautology, one of the hypotheses $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$, or follows by Modus Ponens from two formulas $\mathcal{S}_j$ and $\mathcal{S}_\ell$ that occur earlier in the sequence (that is, $j < i$ and $\ell < i$).

Why are we interested in well formed formulas? For one thing, they are patterns for sentences. For example, " $P \wedge Q$ " is the pattern for all sentences that are compounded with an "and". Why are we interested in demonstrations? Because demonstrations are patterns for arguments. Now, in everyday life, when we present an argument it is because we are trying to convince someone of something. Such an "intended-to-convince" argument is a sequence of sentences, and those sentences are of two kinds: those whose truth is supposed to be clear from an understanding of the facts under discussion, and those whose truth is supposed to be clear on logical grounds. Those sentences whose truth we claim can be determined from factual information correspond to the hypotheses of a demonstration. (The fact that, in a given case, we may be mistaken in this claim really does not concern us at this point. What really interests us here is the form of the argument: if we grant these things we say can be verified from the facts, are we logically compelled to accept the conclusion?) Those sentences whose truth can be determined on logical grounds correspond to the tautologies and Modus Ponens formulas of a demonstration.

If we take a demonstration and replace each propositional variable by a sentence, being careful to replace each occurrence of the same variable by the same sentence, we will obtain a sequence of sentences that trace a line of reasoning from certain hypotheses to a conclusion. Let us call such a sequence of sentences <u>an instance</u> of the original demonstration. As a simple example, recall the first demonstration that we stated which showed that $P \Rightarrow Q,\ P \vdash Q$.

	Demonstration	Program Outline
1)	$P \Rightarrow Q$	Hyp

2) P Hyp

3) Q MP, For 1, For 2

Using the "if...then..." translation of "⊃", and if

P: George is overweight

Q: George must go on a diet

we obtain the instance

4) If George is overweight, then George must go on a diet

5) George is overweight

6) George must go on a diet.

Formulas (1) and (2) are hypotheses of the demonstration, and formula (3) is a Modus Ponens formula, that is, it follows from other formulas by Modus Ponens. Sentences (4) and (5) must be judged true or false on grounds that have nothing to do with the propositional calculus. To decide whether or not George is overweight requires that we evaluate certain facts about George. Even if we agree that George is overweight, we still might reject sentence (4). For example, if George is only one pound overweight, then there's probably no need for him to go on a diet. But, if on examining the facts, we accept sentences (4) and (5) as true, then logic compels us to accept sentence (6) as true.

As the foregoing example illustrates, arguments that are instances of demonstrations are arguments in which we have great confidence. Any rational being would accept as true the concluding sentence of an instance of a demonstration if he or she accepted the hypotheses as true.

If an argument can be modified (in a way which we make precise below) so that it becomes an instance of a demonstration, let us call that argument <u>sound</u>. A sound argument we will call a <u>proof</u>. But here we must warn the reader against thinking that this works the other way around. In Book 2 we will study arguments that we find convincing but which cannot be made into instances of any demonstration in the propositional calculus. We will also wish to call these arguments "sound". Consequently, our view at this point must be that

arguments which can be made into instances of demonstrations are sound arguments of one type. Sound arguments of another type will be studied later.

While we're on the subject of proofs (that is, sound arguments) we must point out that an argument is not guaranteed to be sound simply because someone claims that it is. Human beings do make mistakes. Even very able mathematicians occasionally make a mistake. Indeed there are famous examples of mathematicians who have published arguments in the belief that they were proofs only to have it revealed later that the arguments were not sound. So whenever an argument is presented to you as a proof, you should read it carefully and critically to be sure that the argument is sound. There are more mathematical mistakes in print (perhaps even in this series of books!) than you might imagine. Finding a mistake is part of the fun of reading mathematics.

How do we determine whether an argument is sound? At the moment we have only discussed one way: We know an argument is sound if we can modify it so that it becomes an instance of a demonstration. This modification is achieved by studying the pattern of the argument, restoring hypotheses that have been omitted and reorganizing the argument until it is in the right form for us to check whether or not it is an instance of a demonstration. Let us call this process of modifying an argument in order to check whether it is an instance of a demonstration a Modus Ponens analysis (or simply an analysis) of the argument. To provide an example, let us analyze the argument on page 16 of Book 0, Chapter 1, which we have already considered earlier in this book.

To convince us that $4 \cdot_{29} 24 = 9$, the author argues that, since $96 = 3 \cdot 29 + 9$, 96 is on the fourth winding. There is an unstated hypothesis here, namely, that numbers between $3 \cdot 29$ and $3 \cdot 29 + 28$ inclusive are on the fourth winding. Clearly 96, which is equal to $3 \cdot 29 + 9$, is between $3 \cdot 29$ and $3 \cdot 29 + 28$. Consequently 96 is on the fourth winding. We can summarize this part of the argument in three sentences:

1) If 96 lies between 87 and 115 inclusive, then 96 lies on the fourth winding.

2) 96 lies between 87 and 115 inclusive.

3) 96 lies on the fourth winding.

The argument then continues: The number $3 \cdot 29$ corresponds to 0 on the first winding and hence $3 \cdot 29 + 9$ corresponds to 9 on the first winding. This can be summarized as:

4) If 87 corresponds to 0 on the first winding, then
96 corresponds to 9 on the first winding.

5) 87 corresponds to 0 on the first winding.

6) 96 corresponds to 9 on the first winding.

The argument concludes as follows: Since $4 \cdot 24 = 96$ and 96 corresponds to 9 on the first winding, $4 \cdot_{29} 24 = 9$. We restate this as:

7) $4 \cdot 24 = 96$.

8) $4 \cdot 24 = 96$ and 96 corresponds to 9 on the first winding.

9) If $4 \cdot 24 = 96$ and 96 corresponds to 9 on the first winding, then
$4 \cdot_{29} 24 = 9$.

10) $4 \cdot_{29} 24 = 9$.

We now see that, with

P: 96 lies between 87 and 115 inclusive

Q: 96 lies on the fourth winding

R: 87 corresponds to 0 on the first winding

S: 96 corresponds to 9 on the first winding

T: $4 \cdot 24 = 96$

U: $4 \cdot_{29} 24 = 9$

the argument stated by sentences (1) - (10) outlines an instance of the demonstration outlined below:

11)	$P \Rightarrow Q$	Hyp
12)	P	Hyp
13)	Q	MP, For 11, For 12

14)	$R \Rightarrow S$	Hyp
15)	R	Hyp
16)	S	MP, For 14, For 15
17)	T	Hyp
18)	$T \wedge S$	CI, For 17, For 16
19)	$[T \wedge S] \Rightarrow U$	Hyp
20)	U	MP, For 19, For 18

Consequently, anyone who agrees that in sentences (1) - (10) we have laid out the original argument correctly must now also agree that that argument is sound. A further consequence of this analysis is that, if sentences (1), (2), (4), (5), (7), and (9) are accepted as true, then we are compelled by logic to accept sentence (10) as true.

3.2 The Propositional Calculus and Open Sentences

On page 37 of Book 0, Chapter 2 the author presents an interesting argument intended to convince us that the difference of two integers can be obtained by an appropriate addition. The argument is the following:

For all integers x, y

	1)	$(x-y)+y = x$
But	2)	$(x+{}^{-}y)+y = x+({}^{-}y+y)$
	3)	$= x+0$
	4)	$= x$

The author then concludes that $x-y = x+{}^{-}y$ because there is only one integer which when added to y produces x.

Again we hope you agree with us that this is a convincing argument. But why were we convinced, and of what were we convinced? Earlier we agreed that an argument is a sequence of sentences. A sound argument convinces us of the truth of the concluding sentence <u>if</u> we are convinced of the truth of the hypotheses. But this argument is not a

sequence of sentences; it is a sequence of open sentences, and open sentences we are told are neither true nor false. So how can the argument convince us of anything? Here is one explanation.

An open sentence such as

(1) $(x-y)+y=x$

is a pattern for sentences. If we replace each occurrence of "x" and "y" in (1) by the name of an object of interest, being careful to replace each occurrence of "x" by the same name and each occurrence of "y" by the same name, we obtain a sentence. For example, for x: 2 and y: 3 we obtain

5) $(2-3)+3=2.$

We call (5) an instance of (1). Although "x" and "y" are different variables, the same name can be substituted for both of them. Thus

$$(2-2)+2=2$$

is also an instance of (1). Here are a few more:

$$(6-8)+8=6$$

$$(\hat{5}-8)+8=\hat{5}$$

$$(\hat{5}-\hat{8})+\hat{8}=\hat{5}$$

$$(0-1)+1=0$$

$$(4-0)+0=4$$

The author's introductory phrase "For all integers x, y" tells us that we are only interested in those instances of (1) that can be obtained by replacing "x" and "y" by names of integers.

Each object whose name can replace a variable is called a value for that variable. For the example at hand, each integer is not only a value for "x", but is also a value for "y". Rather than talking about replacing "x" and "y" by names of integers (which is an awkward phrase) we will speak instead of assigning "x" and "y" integral values. For example, if in (1) we assign "x" the value 2 and "y" the value 3, we obtain the instance (5).

We are now ready to discuss the sequence of open sentences (1) - (4). The sequence of open sentences is a pattern for arguments. If we assign "x" and "y" integral values, being careful to assign the same value to each occurrence of the same variable in each of the open sentences (1) - (4), we will obtain an argument. For example, if "x" is assigned the value 2 and "y" the value 3, we obtain:

$$(2 - 3) + 3 = 2$$

But $$(2 + {}^{-}3) + 3 = 2 + ({}^{-}3 + 3)$$

$$= 2 + 0$$

$$= 2 .$$

Consequently,

$$2 - 3 = 2 + {}^{-}3 .$$

We contend that, no matter what integral values are assigned to "x" and "y", the resulting argument will be sound. This being so, it would be convenient if we could say that the original pattern of open sentences (1) - (4), together with the concluding sentence, can be modified until it becomes an instance of a demonstration. But, in order to be able to take this view, we must broaden our notion of "instance" by permitting open sentences to be replacements for propositional variables. This poses no problem. It merely means that, in addition to having sentences such as

George Washington slept here and logic is logical

as instances of the well formed formula " $P \wedge Q$ ", we also have instances such as

$x + y = 0$ and $x > y$

and even

$x + y = 0$ and logic is logical.

With this broader notion of "instance" we can now analyze the part of the argument on page 37 of Book 0, Chapter 2, which is stated by open sentences (2) - (4). The analysis results in the following sequence of open sentences:

6) $(x + {}^{-}y) + y = x + ({}^{-}y + y)$

7) ${}^{-}y + y = 0$

8) $[(x+\bar{}y)+y = x+(\bar{}y+y)] \wedge [\bar{}y+y = 0]$

9) $[(x+\bar{}y)+y = x+(\bar{}y+y) \wedge \bar{}y+y = 0] \Rightarrow [(x+\bar{}y)+y = x+0]$

10) $(x+\bar{}y)+y = x+0$

11) $x+0 = x$

12) $[(x+\bar{}y)+y = x+0] \wedge [x+0 = x]$

13) $[(x+\bar{}y)+y = x+0 \wedge x+0 = x] \Rightarrow [(x+\bar{}y)+y = x]$

14) $(x+\bar{}y)+y = x$

Then it is clear that this outlines an instance of a demonstration, for if we take

P: $(x+\bar{}y)+y = x+(\bar{}y+y)$

Q: $\bar{}y+y = 0$

R: $(x+\bar{}y)+y = x+0$

S: $x+0 = x$

T: $(x+\bar{}y)+y = x$

we have the following demonstration outline:

15)	P	Hyp
16)	Q	Hyp
17)	$P \wedge Q$	CI, For 15, For 16
18)	$[P \wedge Q] \Rightarrow R$	Hyp
19)	R	MP, For 18, For 17
20)	S	Hyp
21)	$R \wedge S$	CI, For 19, For 20
22)	$[R \wedge S] \Rightarrow T$	Hyp
23)	T	MP, For 22, For 21

Notice that in open sentences (6) - (14) above we mixed symbols of the propositional calculus, such as "$\wedge$" and "$\Rightarrow$", with symbols that are not part of the calculus, such as "$\bar{}y+y = 0$". We did so because the meaning is clear, and we already had in mind the demonstration outlined by formulas (15) - (23).

There are several interesting things to observe here. First of all, our nine-step analysis is three times as long as the author's three-step proof. Indeed the author's steps correspond precisely to steps (6), (10), and (14) of our analysis. Does this mean that the author's proof is defective? We claim not. In the biology laboratory students dissect frogs. Dissecting a frog does not imply that there is something wrong with the frog. One dissects a frog in order to learn more about that particular frog and about frogs in general.

In our view the analysis of mathematical arguments such as we are proposing in this book is a kind of dissection. By laying open a given argument we hope to learn more about that argument in particular, and about all arguments in general. One thing the analysis does is to compel all the underlying hypotheses to stand forth and be recognized. In the case before us there are five such hypotheses and they are rather interesting ones. The first, in step (6), is an instance of the associative law of addition for integers. The second, in step (7), is an instance of what might be called "the law of additive inverses". The third and fifth hypotheses occur in steps (9) and (13), respectively. They are instances of the property of equality which is frequently summarized as "equals may be substituted for equals". The fourth hypothesis, in step (11), is an instance of "the law of zero".

We would like to make one more observation about frogs. One cannot dissect a frog without doing serious damage to the frog. In a sense the same is true of analyzing mathematical arguments. When we analyze an argument we force the argument into a standard mold that may not be a natural one. Note, for example, that the above analysis lacks the conciseness, simplicity, and clarity of the author's original proof. The analysis exposes the "innards" of the argument; but to appreciate a good argument (as in the case of a beautiful painting) you must observe it from a proper distance. You must see the total picture and how the whole relates to its essential parts without focusing on details that detract from the global view. It is for this reason that we have made a distinction between "demonstration" and "proof". In actual fact, there is a confusion of language here. The things we have called "demonstrations" are called "proofs" by logicians. On the other hand, the arguments

that demonstrations were designed to mirror are called "proofs" by mathematicians. We propose the mathematical notion of "proof" as the standard you should strive for in your studies and not the demonstrations, even though these latter are so valuable in studying proofs. We have made clear what we mean by a "demonstration". We cannot be quite so clear about what we mean by a "proof". A proof is a sound argument. A good proof should be well organized, interesting, clear, clever, and instructive. A beautiful proof has artistic characteristics that no logical analysis will reveal.

That concludes our sermon for today.

PROBLEM SET 14

1. Write five English sentences that are instances of the well formed formula "$P \wedge [Q \Rightarrow P]$".

2. Write five arithmetic open sentences that are instances of the well formed formula "$P \wedge [Q \Rightarrow P]$".

3. For each of the open sentences you wrote in answer to Problem 2, write two instances, if possible, one true and one false. Assign integral values to all your variables.

4. In our discussion of the argument on page 37 of Book 0, Chapter 2, we used the fact that "there is only one integer which when added to y produces x". Here is an argument which we will show "proves" this statement:

(*)
$$\left\{\begin{array}{l} \text{For all integers } a, x, y, \text{ if } a+y=x, \text{ then} \\ \qquad x+{}^{-}y = (a+y)+{}^{-}y \\ \qquad\qquad = a+(y+{}^{-}y) \\ \qquad\qquad = a+0 \\ \qquad\qquad = a \end{array}\right.$$

Similarly, for all integers b, x, y, if $b+y=x$, then

$$x+{}^{-}y = b.$$

Then, it follows that $a = b$.

We now analyze the part of this argument identified by (*) and in Problem 5 you will show that this analysis outlines an instance of a demonstration. Your task in what follows is to provide any missing information by filling in all the blanks.

1) $a + y = x$

2) $x + \bar{}y = x + \bar{}y$

3) [_____] $\wedge$ [_____]

4) $[a + y = x \wedge x + \bar{}y = x + \bar{}y] \Rightarrow [x + \bar{}y = (a + y) + \bar{}y]$

5) _____

6) $(a + y) + \bar{}y = a + (y + \bar{}y)$

7) _____

8) $[x + \bar{}y = (a + y) + \bar{}y \wedge (a + y) + \bar{}y = a + (y + \bar{}y)] \Rightarrow [x + \bar{}y = a + (y + \bar{}y)]$

9) _____

10) $y + \bar{}y = 0$

11) $[x + \bar{}y = a + (y + \bar{}y)] \wedge [y + \bar{}y = 0]$

12) _____

13) _____

14) $a + 0 = a$

15) $[x + \bar{}y = a + 0] \wedge [a + 0 = a]$

16) _____

17) _____

5. Outline the demonstration of which the argument stated by open sentences (1) - (17) in Problem 4 is an instance, and include a program outline of that demonstration. Hint: You will need to use nine propositional variables; don't forget to indicate which propositional variable is to be replaced by which open sentence in order to produce the given instance.

- o O o -

3.3 The Limitations of the Propositional Calculus

We have developed the propositional calculus as a tool for analyzing mathematical arguments. By the conditions we have set an argument has been successfully analyzed when we can show that it can be modified to become an instance of a demonstration of the propositional calculus. Since the propositional calculus has only one rule of inference, Modus Ponens, it would be a remarkable thing if every mathematical argument were an instance of a demonstration of the propositional calculus. Indeed that would mean that all arguments are Modus Ponens arguments. This however is not the case. In Book 2 we will begin a study of certain arguments that cannot be analyzed by the methods of the propositional calculus. We would like to conclude this book with a brief discussion of some of the issues to be treated in Book 2. To illustrate we offer an example of an argument that requires more than the propositional calculus for its analysis.

Suppose that we wanted to convince someone that the square of every integer is greater than or equal to 0. We might argue in the following way.

We know the following facts about the integers:

(1) For every integer x, $x < 0$ or $x = 0$ or $x > 0$.

(2) The product of two negative integers is positive.

(3) The product of two positive integers is positive.

(4) $0 \cdot 0 = 0$.

From these facts it follows that

(5) If $x < 0$ then $x^2 > 0$

(6) If $x = 0$ then $x^2 = 0$

(7) If $x > 0$ then $x^2 > 0$.

Consequently,

(8) For every integer x, $x^2 > 0$ or $x^2 = 0$ i.e., $x^2 \geq 0$.

We hope you found the argument convincing. Nevertheless, any effort to modify the foregoing argument into an instance of a demonstration of the propositional calculus is

doomed to failure. At this time we will not take up all of the problems that are involved but rather we will point out one specific difficulty.

Consider (1). In spite of the fact that this sentence contains a variable it is not an open sentence. It is a complete declarative sentence as it stands. It says that every integer is negative, zero, or positive. But let us look at the structure of this sentence. We see that it has as a component part

(9) $\qquad x < 0$ or $x = 0$ or $x > 0$.

We recognize (9) as an instance of the well formed formula

$$P \vee Q \vee R \text{ }^{\star}$$

with P: $x < 0$; Q: $x = 0$; and R: $x > 0$. Clearly (9) is an open sentence. But when the phrase "For every integer x" is put in front of the open compound sentence (9) something interesting happens. The result is the sentence (1) which is neither an open sentence nor a compound sentence; that is, (1) is not an instance of any compound well formed formula of the propositional calculus.

In order to analyze sentences like (1) we will introduce a new formal language. We see that for us to be able to analyze (1) we will need this new language to contain the symbols "x", "$<$", "0", "$=$", and "$\vee$". But in addition we will introduce a special symbol "$\forall$" that we call the <u>universal quantifier</u>, with the intent that "$\forall x$" is to play the role of the phrase "For every value of x". Thus (1) will be formalized as

(10) $\qquad (\forall x)[x \in \mathbb{Z} \Rightarrow (x < 0 \vee x = 0 \vee x > 0)]$

which is read "For all x, if x is an integer then $x < 0$ or $x = 0$ or $x > 0$" or "For every integer x, $x < 0 \vee x = 0 \vee x > 0$".

Here is another example. Sentence (8) becomes

(11) $\qquad (\forall x)[x \in \mathbb{Z} \Rightarrow x^2 \geq 0]$.

What do we mean when we say that sentence (11) is true? We mean that every instance of

---

$\star$ Actually this is not a well formed formula. However, "$P \vee [Q \vee R]$" and "$[P \vee Q] \vee R$" are both well formed formulas and $[P \vee [Q \vee R]] \Leftrightarrow [[P \vee Q] \vee R]$. In cases where the placement of the brackets has no effect on the truth value and doesn't lead to any confusion, they are often dropped.

the open sentence "$x \in \mathbb{Z} \Rightarrow x^2 \geq 0$" is true. Here are a few of those instances:

$$0 \in \mathbb{Z} \Rightarrow 0^2 \geq 0$$

$$1 \in \mathbb{Z} \Rightarrow 1^2 \geq 0$$

$$\hat{1} \in \mathbb{Z} \Rightarrow (\hat{1})^2 \geq 0$$

$$2 \in \mathbb{Z} \Rightarrow 2^2 \geq 0$$

$$\hat{2} \in \mathbb{Z} \Rightarrow (\hat{2})^2 \geq 0$$

Universally quantified sentences are not the only ones that cannot be rendered in the propositional calculus. There are other kinds, of which this is one:

(12) There exists an integer x such that $x > 0$ and $2x^2 + 3 = 5$.

In (12) the connecting word "and" focuses our attention on the component part of the sentence

(13) $x > 0$ and $2x^2 + 3 = 5$.

We recognize (13) as an instance of the well formed formula

$$P \wedge Q$$

with P: $x > 0$ and Q: $2x^2 + 3 = 5$. But when the phrase "there exists an integer x such that" is placed in front of the open compound sentence (13) we obtain (12) which is neither open nor compound.

We add to our new language a special symbol "$\exists$", called the existential quantifier, with the intent that "$\exists x$" is to convey the sense of the phrase "there exists an x such that". We then formalize sentence (13) as

(14) $(\exists x)[x \in \mathbb{Z} \wedge x > 0 \wedge 2x^2 + 3 = 5]$,

which is read "there exists an x such that x is an integer and $x > 0$ and $2x^2 + 3 = 5$".

The sentence (14) is true precisely if at least one instance of the open sentence

(15) $x \in \mathbb{Z} \wedge x > 0 \wedge 2x^2 + 3 = 5$

is true. Here are a few instances of (15).

$$0 \in \mathbb{Z} \wedge 0 > 0 \wedge 2 \cdot 0^2 + 3 = 5$$

$$1 \in \mathbb{Z} \wedge 1 > 0 \wedge 2 \cdot 1^2 + 3 = 5$$

$$\hat{1} \in \mathbb{Z} \wedge \hat{1} > 0 \wedge 2 \cdot (\hat{1})^2 + 3 = 5$$

$$2 \in \mathbb{Z} \wedge 2 > 0 \wedge 2 \cdot 2^2 + 3 = 5$$

As we see, most instances of (15) are false; but they are not all false. Since there is at least one true instance of (15), sentence (14) is true.

Let us summarize. Suppose that "S" is an open sentence in "x" containing no other variables. Then

(16) $(\forall x)[S]$

is not an open sentence but a specific statement. That statement is true iff every instance of the open sentence "S" is true.

Similarly,

(17) $(\exists x)[S]$

is also a specific statement that is true iff at least one instance of the open sentence "S" is true. Thus (17) is false precisely if S is false for all values of "x". This fact suggests a link between the two quantifiers that we will now investigate.

Suppose (17) is false. Then every instance of "S" is false and hence every instance of "$\sim$S" is true. Consequently,

(18) $(\forall x)[\sim S]$

is true, and hence the negation of (18)

(19) $\sim[(\forall x)[\sim S]]$

is false. Thus if (17) is false so also is (19).

But the argument can be reversed. Suppose that (19) is false. Then (18) is true and hence every instance of "$\sim$S" is true. But then every instance of "S" must be false and hence (17) is false.

We have proved that (17) is false iff (19) is false. It follows from this that (17) is true iff (19) is true. Consequently

(20) $(\exists x)[S]$ iff $\sim[(\forall x)[\sim S]]$.

Suppose that "S" contains variables other than "x". If each such variable is assigned a value we will obtain a formula "S′" that has no variables in it except "x" and we can repeat the argument above to show that "$(\exists x)[S']$" is true iff "$\sim[(\forall x)[\sim S']]$" is true. Consequently, in this case the two sides of (20) are open sentences, but every time you take instances of these open sentences so that the same replacements are made in each, (20) is true.

By arguments similar to those used above it can be shown that

(21) $(\forall x)[S]$ iff $\sim[(\exists x)[\sim S]]$

is either a true sentence or is true for all "matched" instances of the two component parts. Before we proceed to the problem set, let us consider a few examples.

Example 1: Restate the following in symbolic form:

All human beings are mortal.

We notice that this is a statement about <u>all</u> human beings, and so we will have to make use of the universal quantifier. In order to do that we will have to identify an open sentence in some variable, say "x", and then, using the universal quantifier, make the statement that all instances of that open sentence are true. Clearly, this is easily handled by taking as our open sentence

x is a human being $\Rightarrow$ x is mortal.

Then we can make the required symbolic statement as follows:

$(\forall x)$[x is a human being $\Rightarrow$ x is mortal].

Example 2: Restate the following in symbolic form:

Some dogs do not bark.

Following a similar procedure to that followed in Example 1 we arrive at this symbolic statement

$(\exists x)$[x is a dog $\wedge \sim$[x barks]].

Example 3: Rewrite the following statements in the natural language (English) and, from your knowledge of American hamburger stands, say for each statement whether it is true or false.

i) $(\forall x)$[x is an American hamburger stand $\Rightarrow$ x serves hamburgers]

ii) $(\exists x)$ [x is an American hamburger stand $\wedge$ x serves milk shakes]

iii) $(\exists x)$ [x is an American hamburger stand $\wedge$ x ran for President in 1976]

iv) $(\forall x)$ [x is an American hamburger stand $\Rightarrow$ x is open only to truckers]

v) $(\exists x)$[x is an American hamburger stand $\wedge$ x is open all night]

There are many correct ways of rewriting these statements in good English. We provide only one such way for each statement.

i) Every American hamburger stand serves hamburgers. This is without question a true statement.

ii) Some American hamburger stands serve milk shakes. This is also a true statement.

iii) At least one American hamburger stand ran for President in 1976. This statement is evidently false.

iv) All American hamburger stands are open only to truckers. This is also a false statement.

v) Some American hamburger stands are open all night. This statement is true.

Example 4: For each of the following sentences, make use of (20) or (21) above to help you generate a statement in good English that has the same meaning as the negation of the given sentence.

i) All whole numbers are even.

ii) Some birds cannot fly.

iii) No person likes music.

iv) Some dinosaurs are not chickens.

v) Every logician is a cannibal.

i) All whole numbers are even.

Proceeding as in Example 1 we obtain the following symbolic statement which corresponds to sentence (i):

$$(\forall x)[x \text{ is a whole number} \Rightarrow x \text{ is even}].$$

But from (21) we see that this has the same meaning as

$$\sim[(\exists x)[x \text{ is a whole number} \wedge \sim[x \text{ is even}]]].$$

The negation of this is

$$\sim[\sim[(\exists x)[x \text{ is a whole number} \wedge \sim[x \text{ is even}]]]]$$

which is equivalent to

$$(\exists x)[x \text{ is a whole number} \wedge \sim[x \text{ is even}]].$$

Then, if we make this statement in the natural language (English), we will be stating something that has the same meaning as the negation of sentence (i). One way of making this statement is as follows:

Some whole numbers are not even.

For the remaining sentences we provide an abbreviated discussion only.

ii) Some birds cannot fly.

$$(\exists x)[x \text{ is a bird} \wedge \sim[x \text{ can fly}]]$$

$$\sim[(\forall x)[x \text{ is a bird} \Rightarrow x \text{ can fly}]] \qquad \text{from (20)}$$

$$(\forall x)[x \text{ is a bird} \Rightarrow x \text{ can fly}]$$

is equivalent to the negation of the preceding statement. Hence a statement having the same meaning as the negation of sentence (ii) is

All birds can fly.

iii) No person likes music.

$$(\forall x)[x \text{ is a person} \Rightarrow \sim[x \text{ likes music}]]$$

$$\sim[(\exists x)[x \text{ is a person} \wedge x \text{ likes music}]] \qquad \text{from (21)}$$

$$(\exists x)[x \text{ is a person} \wedge x \text{ likes music}]$$

is equivalent to the negation of the preceding statement. Hence a statement having the

same meaning as the negation of sentence (iii) is

Some people like music.

iv) Some dinosaurs are not chickens.

$(\exists x)[x \text{ is a dinosaur} \wedge \sim[x \text{ is a chicken}]]$

$\sim[(\forall x)[x \text{ is a dinosaur} \Rightarrow x \text{ is a chicken}]$ from (20)

$(\forall x)[x \text{ is a dinosaur} \Rightarrow x \text{ is a chicken}]$

is equivalent to the negation of the preceding statement. Hence a statement having the same meaning as the negation of sentence (iv) is

Every dinosaur is a chicken.

v) Every logician is a cannibal.

$(\forall x)[x \text{ is a logician} \Rightarrow x \text{ is a cannibal}]$

$\sim[(\exists x)[x \text{ is a logician} \wedge \sim[x \text{ is a cannibal}]]$ from (21)

$(\exists x)[x \text{ is a logician} \wedge \sim[x \text{ is a cannibal}]]$

is equivalent to the negation of the preceding statement. Hence a statement having the same meaning as the negation of sentence (v) is

Some logicians are not cannibals.

PROBLEM SET 15

1. Restate each of the following in symbolic form:

   a) All high school freshmen are students.

   b) Some snails eat dinosaurs.

   c) No circles are squares.

   d) All fish have wings.

   e) Some dogs are not gray.

   f) No trees are shrubs.

2. Restate each of the following as a good English sentence:

   a) $(\exists x)[x \text{ is a window} \wedge x \text{ is a hole}]$

   b) $(\forall x)[x \text{ is a window} \Rightarrow \sim[x \text{ is a hole}]]$

   c) $(\forall x)[x \text{ is a window} \Rightarrow [\sim[x \text{ is broken}] \vee x \text{ needs mending}]]$

   d) $\sim[(\forall x)[x \text{ is a window} \Rightarrow [x \text{ is a hole} \Rightarrow x \text{ is broken}]]$

   e) $(\exists x)[x \text{ is a window} \wedge [x \text{ is a hole} \wedge \sim[x \text{ is broken}]]]$

   f) $(\exists x)[x \text{ is a window} \wedge [\sim[x \text{ is broken}] \wedge \sim[x \text{ needs mending}]]]$

3. From your knowledge of the whole numbers, determine the truth value of each of the following statements:

   a) $(\exists x)[x \in W \wedge x+3 = 12]$

   b) $(\forall x)[x \in W \Rightarrow x \geq 0]$

   c) $(\forall x)[x \in W \Rightarrow x^2 + 8 > 12]$

   d) $\sim[(\forall x)[x \in W \Rightarrow 3x - 1 = 11]]$

   e) $\sim[(\exists x)[x \in W \wedge x^2 > 7 \wedge x^2 < 9]]$

   f) $(\forall x)[x \in W \Rightarrow [x \geq 10 \vee x^2 < 87]]$

4. For each of the following, make use of (20) or (21) on pages 150-151 to help you generate a symbolic statement which has the same meaning as the negation of the given statement:

   a) $(\forall x)[x \in W \Rightarrow x+4 = 12]$

   b) $(\exists x)[x \in W \wedge 2x-7 = 6]$

   c) $\sim[(\forall x)[x \in W \Rightarrow 3x = 4]]$

   d) $(\exists x)[x \in W \wedge 3x < 4]$

   e) $(\forall x)[x \in W \Rightarrow \sim[6-x = 5]]$

   f) $(\exists x)[x \in W \wedge 4 = 3 \wedge 2x+1 = 0]$

5. For each of the following sentences, make use of (20) or (21) on pages 150-151 to help you generate a statement in good English that has the same meaning as the negation of the given sentence:

   a) Some dogs purr.

b) No ducks rhumba.

c) All triangles are isosceles.

d) Some students are not here.

e) No triangles are circles.

f) Some people sing.

- o O o -

See Appendix IV for material that will help you review for the test on Chapter 3.

There is much more to be said about quantifiers and their use in mathematics, but we will save that for later. In this book we have introduced the propositional calculus. We have explained its use in analyzing some sorts of mathematical arguments. Now, in this concluding section, we have pointed out the limitations of the propositional calculus and the need for something a little more powerful. We hope you have found Book 1 interesting. If you have, there are better things ahead in Book 2 and in the other books that follow.