IV Sentence Logic: Informal Semantics and Natural Deduction Techniques

1. In the last chapter we constructed a framework, consisting of the apparatus of subordination and reiteration, in which hypothetical reasoning could be displayed. Then we devised a formal system accounting for implication by adding rules to this underlying framework. Just two rules were added, one for introducing the connective '$\supset$' and one for eliminating it.

One nice thing about natural deduction is that these features recur in very tidy and systematic fashion as we expand our logical horizons. If we wish to add more connectives we only have to find out the right pair of rules for the connective; this is almost always an easy thing to do, requiring only a little sensitivity to the way we reason. Once we have got the right pair of rules, we know at once how to incorporate the connective into the logical system.

In this chapter we will be able to dispose quickly of negation and a number of other connectives as well: conjunction, disjunction, and equivalence. By the time we are through, we will have a system of logic that can handle many new sorts of reasoning.

2. Of course, besides adding new rules, we will have to add new formulas as well. A rule for '$\sim$' wouldn't be of much use in a system unless the system allowed formulas involving the connective '$\sim$'. So in all, we will be adding four connectives to the system $S_{\supset}$ of the last chapter: '$\wedge$', '$\vee$', '$\sim$', and '$\equiv$'. Each time we add one of these connectives we will extend the class of formulas with which we are working, and in fact, by the time we finish this chapter, we will have discussed a number of logical systems: $S_{\supset\wedge}$, $S_{\supset\wedge\vee}$, $S_{\supset\wedge\vee\sim}$, and $S_{\supset\wedge\vee\sim\equiv}$.

But it would just be too fussy to have to keep track of things in this way. Rather than discussing at each stage the additions we are making to the set of formulas and giving each of these a name, we will simply regard them as part of a big system. Then we can describe the formulas of that system and be done with it.

The formulas of this inclusive system, S_s, are built up according to the following rules.

1. Any of the sentence parameters of $S_{\supset}$ (see III.5) is a formula of S_s;
2. If A and B are formulas of S_s, then so is $(A \supset B)$;
3. If A is a formula of S_s, then so is $\sim A$;
4. If A and B are formulas of S_s, then so is $(A \wedge B)$;
5. If A and B are formulas of S_s, then so is $(A \vee B)$;
6. If A and B are formulas of S_s, then so is $(A \equiv B)$.

Again, a string of symbols qualifies as a formula of S_s only if it can be shown to be a formula by repeated applications of rules 1 to 6.

Mixing up all these rules at once can lead to the construction of some pretty complicated formulas. The following are examples of formulas of S_s.

$$((p \supset ((q \wedge r) \equiv \sim(s \vee (p \wedge q)))) \vee (r_5 \supset (s \wedge p)))$$
$$((((q \wedge q) \wedge q) \wedge q) \vee p)$$
$$((\sim\sim p_2 \vee \sim\sim q_1) \equiv (p \wedge q_5))$$
$$(\sim p \supset ((q \wedge q) \vee \sim(r \wedge s)))$$

In the sections below we will deal in turn with each of these new connectives. The plan will be (1) to discuss briefly the translation of sentences of natural language into formulas involving the connective, (2) to discuss specimen arguments using English equivalents of the connective, and (3) to use these arguments in figuring out introduction and elimination rules for the connective.

3. In this section we will consider conjunction, which means that on the formal side of things we will be dealing with the connective '$\wedge$', and on the informal side with the word 'and'. The sentence 'Brasidas was a great general, and Thucydides was a great historian' would be translated into S_s as '$p \wedge q$', letting 'p' and 'q' stand for the conjuncts of the English sentence. Similarly, 'Chicago and New York are large cities' would appear as a formula like '$p \wedge q$'; but here one must be sensitive to nuances. 'Jack and Jill went up the hill', or better, 'Jack and Jill got married' conveys something more than the conjunction in which we are interested for logical purposes; there is the suggestion that the acts are done together in the first case, and reciprocally in the second. 'Jack counted three and pulled the ripcord', on the other hand, suggests that the actions were performed in a certain order; this also is a feature not taken into consideration in logical conjunction.

In working out logical rules for conjunction, our problem is to decide how we reason to conjunctions in arguments, and how we reason from them. As it turns out, this is a very simple matter; if you think about it a minute you should be able to work out the answer.

Suppose, for instance, that we want to show that the square root of 5 is less than 3 and greater than 2. Well, we would have to show two things: that $\sqrt{5}$ is less than 3 and that $\sqrt{5}$ is greater than 2. (It doesn't matter in which order we do this, as long as we get them both.)

So, we might argue as follows. (To keep from digressing too far, we'll suppose that for all real numbers x and y, $x < y$ if and only if $x^2 < y^2$—a real number is less than another just in case its square is less than the other's square.)

1. $3^2 = 9$.
2. $(\sqrt{5})^2 = 5$.
3. $5 < 9$.
4. So $(\sqrt{5})^2 < 3^2$, in view of 1, 2, and 3.
5. But then $\sqrt{5} < 3$.
6. $2^2 = 4$.
7. $4 < 5$.
8. So $2^2 < (\sqrt{5})^2$, in view of 2, 6, and 7.
9. But then $2 < \sqrt{5}$.
10. Therefore, in view of 5 and 9, $\sqrt{5} < 3$ and $2 < \sqrt{5}$.

(i)

The step in which we are interested here is the last, number 10. If we translate the inference used in passing from 6 and 9 to 10 into the notation of S_s, we get something like this.

$$\begin{array}{|l} \vdots \\ p \\ \vdots \\ q \\ \vdots \\ (p \wedge q) \end{array}$$

(ii)

And now it's easy to state in general the introduction rule for '$\wedge$' in S_s; we have to be careful to remember, though, that the order in which we got the premisses in i didn't matter; we could as well have proved that $2 < \sqrt{5}$ before proving that $\sqrt{5} < 3$. Now for the rule; applications of the rule of *conjunction introduction* (*conj int*) are permitted in the system S_s, as follows.

$$\begin{array}{|l} \vdots \\ A \\ \vdots \\ B \\ \vdots \\ (A \wedge B) \end{array} \qquad \begin{array}{|l} \vdots \\ B \\ \vdots \\ A \\ \vdots \\ (A \wedge B) \end{array}$$

That is, whenever we have written down two formulas *A* and *B* (in any order) in a derivation in S_s, we may then write down in that derivation the result $(A \wedge B)$ of conjoining *A* with *B*.

4. If anything, it's easier to grasp the rule of conjunction elimination, and to avoid being long-winded we may as well state it without any fanfare. This rule is applied as follows.

$$\begin{array}{|l} \vdots \\ (A \wedge B) \\ \vdots \\ A \end{array} \qquad \begin{array}{|l} \vdots \\ (A \wedge B) \\ \vdots \\ B \end{array}$$

Whenever we have written a conjunction $(A \wedge B)$ in a derivation, we may thereafter write down either of its conjuncts, *A* and *B*.

So far then, we have discussed two pairs of rules for logical connectives of the system S_s: the connectives '$\supset$' and '$\wedge$'. Here are some specimen derivations making use of these rules.

1	p	hyp
2	q	hyp
3	p	1, reit
4	$(p \wedge q)$	2, 3, conj int
5	$(q \supset (p \wedge q))$	2–4, imp int
6	$(p \supset (q \supset (p \wedge q)))$	1–5, imp int

(iii)

1	$(p \wedge (q \wedge r))$	hyp
2	p	1, conj elim
3	$(q \wedge r)$	1, conj elim
4	q	3, conj elim
5	$(p \wedge q)$	2, 4, conj int
6	r	3, conj elim
7	$((p \wedge q) \wedge r)$	5, 6, conj int

(iv)

5. Now for disjunction, expressed often by the English 'or'. The connective '$\vee$' of S_s corresponds to the inclusive sense of 'or', in which the alternative in which both disjuncts hold true is not meant to be excluded.

For instance, if Smith maintains that Jones is a miser or a vindictive troublemaker, we would not say that Smith was wrong in case both turn out to be true. Smith may not have supposed that both in fact were true, or he would not have made his point so guardedly; he would have said that Jones was a miser *and* a vindictive troublemaker. But still, the disjunctive claim is true under these circumstances. This inclusive sense of 'or' is sometimes expressed by 'and/or' in circumstances (for instance those in which one is formulating rules of some sort) where the speaker feels it is important to make his meaning very precise.

The English 'unless' also has an inclusive sense which would be properly translated in S_s by uses of '$\vee$'. For instance 'Unless my memory is bad the sun is 93 million miles from the earth' could be translated by '$(p \vee q)$', where 'p' stands for 'My memory is bad' and 'q' for 'The sun is 93 million miles from the earth'. In this example, the first claim would still hold good in case 'My memory is bad' and 'The sun is 93 million miles from the earth' are both true; I may have made a lucky guess.

There are difficulties with 'unless' which make it risky to translate English sentences involving this word into S_s using '$\vee$'. Briefly, the trouble is that

S_s will sanction the inference of '(q ∨ p)' from '(p ∨ q)'. But this inference is not valid for 'unless'; for instance, it would be nonsense to claim that 'Unless you will see the stars tonight, it is cloudy' is true if 'Unless it will be cloudy, you will see the stars tonight' is. I would say that this problem indicates a limitation of S_s, that cannot be cleared up without constructing a formal theory of subjunctive conditionals (see II.3, example xxiv). But that's another story.

Bearing in mind the point that disjunction in S_s is inclusive, let's consider a few more examples. Sentences such as 'Alaska or Texas is the largest state in the union' and 'It will rain tonight or I'm a monkey's uncle' would be translated in S_s by sentences of the form '(p ∨ q)'. But we ought to be a bit hesitant about translating something like 'My son will mow the lawn or I'll take away his teddy bear' in this way, since it connotes that it will not be the case that both disjuncts hold true.

There are many examples in which the joint case simply does not come up: for instance, 'He is in Boston or Zagreb', or 'Either he beats his wife or he doesn't'. In both of these, the situation in which both disjuncts are true is excluded out of hand, and it doesn't seem a matter of great importance whether or not we say these sentences are true in case both of their disjuncts are. For this reason, there seems to be no harm in translating these into S_s by formulas of the sort 'p ∨ q'.

6. Let's turn now to the question of logical rules for disjunction. In this case, it's more convenient to take up the elimination rule first. The rule of disjunction elimination is more complicated than the rules we have discussed up to now; but it corresponds closely to the way we actually reason from disjunctions, and again is made to seem natural by considering examples of this sort of reasoning. Let's consider a case in which we must argue from a disjunction to some conclusion; for instance, suppose we want to show that if a number n is greater than 20 or less than 10, then $(n - 15)^2$ is greater than 25. The thing to do is to reason as follows.

1. Assume that $n > 20$ or $n < 10$.
2. Suppose that $n > 20$.
3. Then $(n - 15) > 5$, so $(n - 15)^2 > 25$.
4. On the other hand, suppose that $n < 10$.
5. Then $(n - 15) < -5$.
6. So in this case also, $(n - 15)^2 > 25$.
7. Therefore, $(n - 15)^2 > 25$.
8. Hence, if $n > 20$ or $n < 10$, then $(n - 15)^2 > 25$.

(*v*)

Translating the key steps of this argument into S_s and putting the argument into the format of subordinated derivations, we get something like this.

```
|  |_(p ∨ q)
|  |   |_p
|  |   |  :
|  |   |  r
|  |
|  |   |_q
|  |   |  :
|  |   |  r
|  |  r
|  ((p ∨ q) ⊃ r)
```

(*vi*)

Here, 'p' stands for '$n > 20$', 'q' for '$n < 10$', and 'r' for '$(n - 15)^2 > 25$'. Example vi gives us a pretty good picture of how to get a conclusion from a disjunction '(p ∨ q)'. Besides '(p ∨ q)' we need two other premisses in order to extract a conclusion 'r': a derivation of 'r' from 'p' and one of 'r' from 'q'.

In general, the rule of *disjunction elimination* (*dis elim*) looks like this.

```
| :              | :              | :
| (A ∨ B)        | (A ∨ B)        |   |_B
| :              | :              |   |  :
|   |_A          |   |_B          |   |  C
|   |  :         |   |  :         | :
|   |  C         |   |  C         | (A ∨ B)          etc.
| :              | :              | :
|   |_B          |   |_A          |   |_A
|   |  :         |   |  :         |   |  :
|   |  C         |   |  C         |   |  C
| :              | :              | :
| C              | C              | C
```

That is, whenever we have obtained as items in a derivation a formula $(A \vee B)$ and derivations of C from A and of C from B (in any order), we may then write down C. Above, all the different possible arrangements aren't displayed; there are six in all.

Notice, by the way, that the rule of dis elim makes good sense from the standpoint of truth. If $(A \vee B)$ is true, then A is true or B is true, and so if C follows from A and follows from B, then C is true if A is, and also true if B is.

So C is true in either case. We will develop this idea in great detail below in Chapter VI, where we discuss the semantic interpretation of S_s.

The rule of disjunction elimination, by the way, corresponds to one of the favorite devices of classical orators. This is to discuss a question by beginning with two alternatives, and then giving a pair of speeches designed to show that in either case the desired conclusion follows. Socrates' speech about death in the *Apology* is an example: he asserts that death is either a form of unconsciousness or of consciousness and then argues that in either case it is a good.

7. It isn't easy to motivate the rule of disjunction introduction in the same way we have developed previous rules by referring to samples of reasoning. This is because we hardly ever use this rule in actual argumentation; after we have worked out the rule you will see why this is so.

However, we can get at the rule in a slightly different way by talking about the *justification* of a disjunctive claim. If Peterson asserts that Scrags' car has a short in the generator or a broken fanbelt, how would we go about justifying this assertion? Well, we will have determined that Peterson is right as soon as we have verified that the car has a short in the generator, or verified that it has a broken fanbelt. We know that a disjunctive claim is true as soon as we know that either of its disjuncts is true.

Surely, the rule sanctioning this sort of justification is one which permits the inference of a disjunction from either of its disjuncts: for instance, 'Scrags' car has a short in the generator or a broken fanbelt' from 'Scrags' car has a short in the generator', and 'Scrags' car has a short in the generator or a broken fanbelt' from 'Scrags' car has a broken fanbelt'.

The formal counterpart in S_s of this rule will permit us to argue to disjunctions $(A \vee B)$; this rule of *disjunction introduction* (*dis int*) takes the following form.

⋮	⋮
A	B
⋮	⋮
$(A \vee B)$	$(A \vee B)$

Whenever in a derivation we have written either A or B, we may thereafter write the disjunction $(A \vee B)$.

Now that we have the rule, we can go back to the point with which we began this section and explain why the rule is not often used explicitly in argumentation. The reason is that if we already *know* a thing—say, that the car has a broken fanbelt—we wouldn't bother to reason from this to a disjunction—say, that the car has a short in the generator or a broken fanbelt. We would only lose information in doing this; it is silly, unless one is dissimulating, to assert a disjunction when one knows which of the two disjuncts is true.

Nevertheless, we have shown that dis int is important in the justification of disjunctive statements, and it turns out to be just the right rule for formal purposes. As we will see later, it generates, together with the other rules, a theory of disjunction which is *semantically complete*.

8. At this point we have, besides reit, six rules with which to build derivations. (Two rules apiece for each of the connectives '⊃', '∧', and '∨'; remember that 'implication elimination' is just another name for *modus ponens*.) Here are some examples.

```
 1   |_(p ∨ (p ∧ r))          hyp
 2   |  |_p                   hyp
     |  |
 3   |  |_(p ∧ r)             hyp
 4   |  |  p                  3, conj elim
 5   | p                      1, 2, 3–4, dis elim
```

(*vii*)

```
 1   |_((p ∨ q) ⊃ r)          hyp
 2   |  |_p                   hyp
 3   |  | (p ∨ q)             2, dis int
 4   |  | ((p ∨ q) ⊃ r)       1, reit
 5   |  | r                   3, 4, m p
 6   | (p ⊃ r)                2–5, imp int
```

(*viii*)

```
 1   |_(p ∨ (q ∧ r))              hyp
 2   |  |_p                       hyp
 3   |  | (p ∨ q)                 2, dis int
 4   |  | (p ∨ r)                 2, dis int
 5   |  | ((p ∨ q) ∧ (p ∨ r))     3, 4, conj int
     |
 6   |  |_(q ∧ r)                 hyp
 7   |  | q                       6, conj elim
 8   |  | (p ∨ q)                 7, dis int
 9   |  | r                       6, conj elim
10   |  | (p ∨ r)                 9, dis int
11   |  | ((p ∨ q) ∧ (p ∨ r))     8, 10, conj int
12   | ((p ∨ q) ∧ (p ∨ r))        1, 2–5, 6–11, dis elim
```

(*ix*)

1	$((p \vee q) \wedge (q \supset r))$	hyp
2	$(q \supset r)$	1, conj elim
3	$(p \vee q)$	1, conj elim
4	p	hyp
5	$(p \vee r)$	4, dis int
6	q	hyp
7	$(q \supset r)$	2, reit
8	r	6, 7, m p
9	$(p \vee r)$	8, dis int
10	$(p \vee r)$	3, 4–5, 6–9, dis elim

(*x*)

1	$(p \wedge ((p \supset q) \vee r))$	hyp
2	p	1, conj elim
3	$((p \supset q) \vee r)$	1, conj elim
4	$(p \supset q)$	hyp
5	p	2, reit
6	q	4, 5, m p
7	$(q \vee r)$	6, dis int
8	r	hyp
9	$(q \vee r)$	8, dis int
10	$(q \vee r)$	3, 4–7, 8–9, dis elim

(*xi*)

9. Now we come to negation. We have, of course, already dealt with proofs involving '$\sim$' in the system S_0 of Chapter I, and have discussed in Chapter II the translation of negative English sentences into formal notation. Let's turn at once, then, to the problem of reproducing in S_s the way in which negation figures in correct reasoning.

The rule of negation introduction is fairly straightforward. Suppose that we want to prove something negative: say, that the square root of 2 is irrational. This means that there are no whole numbers n and m such that $\sqrt{2} = n/m$. This is to say, $\sqrt{2}$ is not a fraction, or *ratio*.

The following proof of the irrationality of $\sqrt{2}$ was known in antiquity.

1. Suppose that $\sqrt{2}$ is rational.
2. Then for some whole numbers n and m, $\sqrt{2} = n/m$.

3. By factoring out all the common divisors of n and m, we obtain whole numbers j and k having no common divisors but 1, such that $^{j}/_{k} = {}^{n}/_{m}$.
4. In view of step 3, $^{j}/_{k}$ is in lowest terms.
5. Since $\sqrt{2} = {}^{j}/_{k}$, $2 = {}^{j^2}/_{k^2}$.
6. Therefore $2 \cdot k^2 = j^2$, so that j^2 is divisible by 2.
7. But if 2 divides j^2 (i.e., divides $j \cdot j$), then 2 is a factor of j, and so j is even.
8. Since j is even, 4 is a factor of j^2.
9. But then, since $2 \cdot k^2 = j^2$, 2 is a factor of k.
10. Therefore k and j are both divisible by 2.
11. So $^{j}/_{k}$ is not in lowest terms, which contradicts step 4.
12. Therefore $\sqrt{2}$ is not rational.

(*xii*)

In the last step of xii, a negative conclusion has been inferred from steps 1 to 11. For our purposes, the crucial features of the reasoning are steps 1, 4, and 11; the remaining steps are mathematical stages in the extraction of a contradiction from the assumption that $\sqrt{2}$ is rational.

Translating these main steps into S_s and casting the argument into subordinated form we obtain the following pattern.

```
| |_p
| | :
| | q
| | :
| | ~q
| ~p
```

(*xiii*)

This rendering of xii displays the rule of neg int perspicuously enough so that it is a simple matter to extract the general rule. Again, however, we have to remember that in xiii we could as well have gotten '~q' before 'q'.

```
| :              | :
| |_A            | |_A
| | :            | | :
| | B            | | ~B
| | :            | | :
| | ~B           | | B
| :              | :
| ~A             | ~A
```

Whenever we obtain as an item in a derivation a subordinated derivation of two contradictory formulas B and $\sim B$ (in either order) from a hypothesis A, we may conclude $\sim A$ in the original derivation.

You may already know the traditional name of this rule: *reductio ad absurdum*. Certainly, you have used arguments which are instances of this rule. Neg int is one of the most frequently used argument patterns in everyday reasoning; in any situation in which the object is to show that something is not the case, the natural thing to do is to assume it and attempt to derive an absurdity. In ordinary cases, this absurdity is not a flat contradiction, as we have required in S_s; it often is a conclusion contrary to popular opinion, or perhaps to the accepted tenets of some profession. We can handle arguments of this sort in S_s by treating these opinions, tenets, or whatever as hypotheses of the whole argument. They would then be reiterated to produce an out-and-out contradiction: that is, a formula together with its negation.

10. The rule of negation elimination raises questions which are more ticklish than any we have encountered so far in this section. It is possible to disagree radically concerning this rule, and the decision made at this juncture can lead to various different logics. Nor are all of these options—in particular, the so-called intuitionistic and classical logics—mere formal games. Both have their appropriate philosophical justification, and both are founded in patterns of reasoning that actually are employed.

An examination of these issues thorough enough to give satisfaction would carry us far astray from the task of presenting the fundamental techniques of modern logic. So we will ride roughshod over the subtleties, and just present the classical approach. Once the techniques are out in the open, they can be applied as readily to intuitionistic as to classical systems of logic, and it will be more easy to appreciate subtleties. For the time being, though, keep in mind that there are legitimate alternatives to the course we will take.

11. To isolate a rule of negation elimination that will generate the classical logic, let's go back to *reductio ad absurdum* arguments. There is a frequently used type of argument which uses the device of reduction to absurdity and yet does not fit the pattern of negation elimination.

I am thinking here of cases in which the conclusion of the argument is not a negative statement. Consider, for instance, the following proof by *reductio* that every real number is equal to, greater than, or less than zero.

1. Suppose that it is not the case that for every real number x, $x = 0$ or $x > 0$ or $x < 0$.

2. Then for some real number, say r, it is not the case that $r = 0$ or $r > 0$ or $r < 0$.
3. Therefore not $r = 0$ and not $r > 0$ and not $r < 0$.
4. But since not $r = 0$ and not $r > 0$, then $r < 0$.
5. And in view of step 3, not $r < 0$; which contradicts step 4.
6. Therefore for every real number x, $x = 0$ or $x > 0$ or $x < 0$.

(xiv)

From the mathematical standpoint this argument is superficial, but it illustrates our point. Rendering the most important parts of xiv (namely, steps 1, 4, 5, and 6) in S_s, we get something looking like this.

```
| |_~p
| | ⋮
| | q
| | ~q
| p
```

(xv)

Example xv is certainly reminiscent of the rule of negation introduction, but if you look back at this rule you will see that xv is *not* an instance of neg int. Now, can we *derive* the inference xv using neg int and the other rules of S_s? That is, can we use the rules we have discussed so far to get to 'p' from a derivation of 'q' and of '~q' from '~p'? You may want to try before reading on.

If you did, I hope you didn't succeed! Example xv cannot be derived from our present stock of rules. (I say this, though it isn't as easy as you might think to find a convincing proof that this is so.) The closest we can come is this.

```
| |_~p
| | ⋮
| | q
| | ~q
| ~~p
```

(xvi)

In xvi, '~~p' is justified by the rule of neg int. But there is no way to get from '~~p' to 'p', our desired conclusion.

By now, you must have guessed what the rule of neg elim will be: it is precisely what we need in this situation, and can be pictured as follows.

$$\begin{array}{|l} \vdots \\ \sim\sim A \\ \vdots \\ A \end{array}$$

The rule of neg elim permits one to write down A in any derivation in which a step $\sim\sim A$ has appeared.

12. Using the rules of neg int and neg elim in connection with the other rules discussed above, we can build derivations of considerable complexity. Adding rules for negation, in fact, makes things much less straightforward than they were before; it often is necessary to be devious and indirect in finding derivations, as in the categorical derivation (example xviii) of '(p ∨ ∼p)' below.

For this reason we will provide in this section a number of examples of derivations of various kinds. In all but two of these examples the reasons are omitted; filling the rest in is left as an exercise.

1	p	hyp
2	∼p	hyp
3	∼q	hyp
4	p	1, reit
5	∼p	2, reit
6	∼∼q	3–5, neg int
7	q	6, neg elim

(xvii)

The next derivation shows that our system conforms to the law of excluded middle: '(p ∨ ∼p)' can be derived categorically in S_s. The laws of excluded middle and noncontradiction are among the first principles of logic to have been formulated. According to the principle of excluded middle any sentence such as 'It is raining this afternoon or it isn't' which corresponds to the formula '(p ∨ ∼p)' must be true. According to the principle of non-contradiction any sentence such as 'It is raining this afternoon and it isn't' which corresponds to the formula '(p ∧ ∼p)' must be false.

Like many derivations in S_s with disjunctive conclusions, the argument is indirect. It would be impossible to obtain '(p ∨ ∼p)' in a categorical derivation by the rule dis int, since neither 'p' or '∼p' can be derived categorically in S_s. (We can't prove this yet, but it's so anyway.) The only way to carry out the derivation is to obtain '∼∼(p ∨ ∼p)' using neg int and the rules for disjunction; then '(p ∨ ∼p)' can be obtained by neg elim.

```
1   | | ~(p ∨ ~p)            hyp
    | |-----
2   | | | p                  hyp
    | | |---
3   | | | (p ∨ ~p)           2, dis int
4   | | | ~(p ∨ ~p)          1, reit
5   | | ~p                   2–4, neg int
6   | | (p ∨ ~p)             5, dis int
7   | ~~(p ∨ ~p)             1–6, neg int
8   | (p ∨ ~p)               7, neg elim
```

(*xviii*)

The derivations below use the rules of S_s (except for the rules for '≡', which we've not yet discussed) in various combinations.

```
| ~(p ∨ q)
|-----
| | p
| |---
| | (p ∨ q)
| | ~(p ∨ q)
| ~p
| | q
| |---
| | (p ∨ q)
| | ~(p ∨ q)
| ~q
| (~p ∧ ~q)
```

(*xix*)

```
| (~p ∨ q)
|-----
| | p
| |---
| | (~p ∨ q)
| | | ~p
| | |---
| | | | ~q
| | | |---
| | | | p
| | | | ~p
| | | ~~q
| | | q
| |
| | | q
| | |---
| |
| | q
| (p ⊃ q)
```

(*xx*)

```
| | ~~(p ∧ ~p)
| |-----
| | (p ∧ ~p)
| | p
| | ~p
| ~~~(p ∧ ~p)
```

(*xxi*)

```
| p
| ~q
|-----
| | (p ⊃ q)
| |---
| | p
| | q
| | ~q
| ~(p ⊃ q)
```

(*xxii*)

```
| |_(p ⊃ ~q)
| | |_q
| | | |_p
| | | | (p ⊃ ~q)
| | | | ~q
| | | | q
| | | ~p
| | (q ⊃ ~p)
| ((p ⊃ ~q) ⊃ (q ⊃ ~p))
```
(*xxiii*)

```
|_~(p ⊃ q)
| |_q
| | |_p
| | | q
| | (p ⊃ q)
| | ~(p ⊃ q)
| ~q
```
(*xxiv*)

13. The only connective of S_s which we've not yet discussed is equivalence or, as it is sometimes called, the biconditional. This is often expressed in English by 'if and only if', or by locutions using phrases such as 'necessary and sufficient condition'. For instance, 'The senate will pass the bill if and only if the house of representatives does', and 'This polygon has three sides if and only if it has three angles' would both be rendered in S_s by formulas of the sort '$(p \equiv q)$'.

At this point there probably is no need to produce examples of arguments to and from biconditional statements. All that is needed to work out the rules for '$\equiv$' is the thought that statements such as 'Harrison is the mayor if and only if he has been duly elected' amount to conjunctions of implicative statements—in this case, to the conjunction 'Harrison is the mayor if he has been duly elected, and Harrison is the mayor only if he has been duly elected'; i.e., to the conjunction 'If he has been duly elected then Harrison is the mayor, and if Harrison is the mayor then he has been duly elected'.

The rules for '$\equiv$', then, will be doubled versions of the rules for '$\supset$'. Thus the rule of equivalence introduction takes the following form.

```
| ⋮              | ⋮
| |_A            | |_B
| | ⋮            | | ⋮
| | B            | | A
| ⋮              | ⋮
| |_B            | |_A
| | ⋮            | | ⋮
| | A            | | B
| ⋮              | ⋮
| (A ≡ B)        | (A ≡ B)
```

Whenever we have obtained derivations of A from B and of B from A (in either order) as items in another derivation, we may thereafter write down $(A \equiv B)$.

And the rule of equivalence elimination is like *modus ponens*, although it is symmetrical.

```
| ⋮          | ⋮          | ⋮          | ⋮
| (A ≡ B)    | (A ≡ B)    | A          | B
| ⋮          | ⋮          | ⋮          | ⋮
| A          | B          | (A ≡ B)    | (A ≡ B)
| ⋮          | ⋮          | ⋮          | ⋮
| B          | A          | B          | A
```

Whenever we have obtained both $(A \equiv B)$ and A (in any order) in a derivation, we may conclude B; and whenever we have obtained both $(A \equiv B)$ and B (in any order) in a derivation, we may conclude A.

14. Below are some examples of derivations using the rules of eqv int and eqv elim, together with the other rules of S_s.

```
 1  | |_(p ⊃ q)                    hyp
 2  | |  |_~q                      hyp
 3  | |  |  |_p                    hyp
 4  | |  |  | (p ⊃ q)             1, reit
 5  | |  |  | q                    3, 4, m p
 6  | |  |  | ~q                   2, reit
 7  | |  | ~p                      3–6, neg int
 8  | | (~q ⊃ ~p)                  2–7, imp int

 9  | |_(~q ⊃ ~p)                  hyp
10  | |  |_p                       hyp
11  | |  |  |_~q                   hyp
12  | |  |  | (~q ⊃ ~p)           9, reit
13  | |  |  | ~p                   11, 12, m p
14  | |  |  | p                    10, reit
15  | |  | ~~q                     11–14, neg int
16  | |  | q                       15, neg elim
17  | | (p ⊃ q)                    10–16, imp int
18  | ((p ⊃ q) ≡ (~q ⊃ ~p))        1–8, 9–17, eqv int
```

(xxv)

```
1   |_(p ≡ (q ≡ q))          hyp
2   |  |_q                   hyp
    |  |
3   | (q ≡ q)                2, eqv int
4   | p                      1, 3, eqv elim
```

(xxvi)

In this last example, step 3 suffices to introduce '$(q \equiv q)$' by eqv int, because it is not required by this rule that the derivations of *B* from *A* and of *A* from *B* needed to introduce $(A \equiv B)$ must be distinct. In this particular case *A* is the same formula as *B* and there is just one derivation, consisting only of step 2, but this is enough to justify step 3.

```
| (p ≡ q)
|_(r ≡ s)
|  |_(p ⊃ r)
|  |  |_q
|  |  | (p ≡ q)
|  |  | p
|  |  | (p ⊃ r)
|  |  | r
|  |  | (r ≡ s)
|  |  | s
|  | (q ⊃ s)
|
|  |_(q ⊃ s)
|  |  |_p
|  |  | (p ≡ q)
|  |  | q
|  |  | (q ⊃ s)
|  |  | s
|  |  | (r ≡ s)
|  |  | r
|  | (p ⊃ r)
| ((p ⊃ r) ≡ (q ⊃ s))
```

(xxvii)

```
|_(p ≡ q)
|  |_(p ∧ ~q)
|  | p
|  | ~q
|  | (p ≡ q)
|  | q
| ~(p ∧ ~q)
|  |_(~p ∧ q)
|  | ~p
|  | q
|  | (p ≡ q)
|  | p
| ~(~p ∧ q)
| (~(p ∧ ~q) ∧ ~(~p ∧ q))
```

(xxviii)

15. Before turning to other matters, let's think a moment about how to find derivations in S_S. In this area nothing can substitute for practice, and in working on the exercises you may already have discovered for yourself some of the tricks discussed below. But maybe it will be helpful to make them explicit here.

First, there are some rules that can be very useful in keeping on the right track when trying to construct a derivation. They can never get you into trouble and often will keep you out of blind alleys.

1. When trying to derive a formula of the sort $(A \supset B)$, always set up a subordinate derivation with hypothesis A and try to get B.
2. Similarly, when trying to derive $\sim A$, set up a subordinate derivation with hypotheses A, and try to get a contradiction.
3. Do the same with formulas of the sort $(A \wedge B)$ and $(A \equiv B)$; that is, try to get them by the appropriate introduction rules.
4. Be careful about getting formulas of the sort $(A \vee B)$ directly, by dis int. Often you have to get it from $\sim\sim(A \vee B)$ by neg elim, or by dis elim or some other elimination rule. See, for instance, example xviii and the paragraphs preceding it.
5. Whenever you have a premiss of the sort $(A \vee B)$, use dis elim unless there is some obvious other way of getting the desired conclusion.
6. Whenever there is an obvious application of an elimination rule—for instance, if you have two formulas A and $(A \supset B)$, or a formula $(A \wedge B)$—use the elimination rule unless you see some obvious other way to get the desired conclusion.
7. In general, work from the top and bottom of an incomplete derivation in towards the middle, using rules 1 to 6 wherever applicable. This means, roughly, using elimination rules to argue from the formulas given at the top and introduction rules to get the formulas desired at the bottom. If the problem is solvable, you often will see at once how to fill in the middle to obtain the desired derivation. But sometimes ingenuity is required, even after using rules 1 to 6 as much as they can be used. Practice is the only thing that can develop real skill in such situations.

We've called the above admonitions "rules", but you must realize that they aren't *rules of* the system S_s. The latter sort of rules, if you like, are rules for *recognizing* (or *characterizing*) derivations; the former are rules of thumb for *finding* derivations. It is like the difference between rules of chess like 'Try not to lose your queen' and 'The king may move one square in any direction, unless the square is threatened by an opposing piece'. The latter is one of the rules constituting the game of chess; the former is a piece of advice intended to help a player do well.

At the risk of making this analogy tedious, let's consider one more point.

To play chess at all well, you have to think ahead—and this requires the assimilation of certain recurring patterns. You learn that if you take a piece that is defended, the attacking piece will usually be taken; for instance, if you take a defended knight with a bishop, the bishop will ordinarily be taken itself. Stringing together such patterns, one is able to extrapolate a position into the future and weigh possible alternatives with more skill. Very good chess players learn by rote large numbers of classical openings, many of them quite long and complicated.

Actually, this sort of thing is not peculiar to games—it takes place in learning any skill. Basic steps cluster into patterns, and these in turn form into larger groups. And in learning to find derivations in S_s, the sort of patterns one has to learn are *argument forms*. For instance, example xvii establishes an important such pattern: whenever you have contradictory formulas A and $\sim A$ in a derivation you can use the rules of proof of S_s to obtain any formula B whatsoever. The strategy is this.

```
| ⋮              | ⋮
| A              | ~A
| ⋮              | ⋮
| ~A             | A
| ⋮              | ⋮
|  |_~B          |  |_~B
|  |  A          |  |  A
|  |  ~A         |  |  ~A
| ~~B            | ~~B
| B              | B
```

(*xxix*)

The above display shows in general how to go about getting *any* formula B, once one has obtained *any* contradiction. In a way, xxix establishes a rule of *contradiction elimination*: from A and $\sim A$ to infer B. But this is not a rule of S_s in the sense that, say, neg elim is a rule of that system. If, for instance, we were simply to proceed as follows,

```
|_p
|  |_~p
|  |  p
|  |  q
```

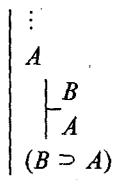
(*xxx*)

this would *not* be a derivation in S_s. But xxix shows us how to fill in steps in xxx to make a derivation in S_s out of it.

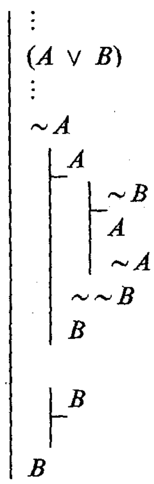
Rules such as neg elim are often called *primitive*, and rules such as contradiction elimination *derived.* The former rule is a primitive rule of S_s, in that it is a basic or unanalyzable principle of proof in the system. The latter rule is derivable in S_s, because the primitive rules of S_s may be manipulated in such a way as to justify any instance of contradiction elimination, as is shown by the scheme xxix. Derived rules, then, are justified by argument patterns which are complexes of primitive rules.

Notice, by the way, that the notions of *primitive* and *derived rule* are relative to the system under consideration. Contradiction elimination might well be a primitive rule of some other system.

Here are two more argument patterns, which often are useful in finding derivations.



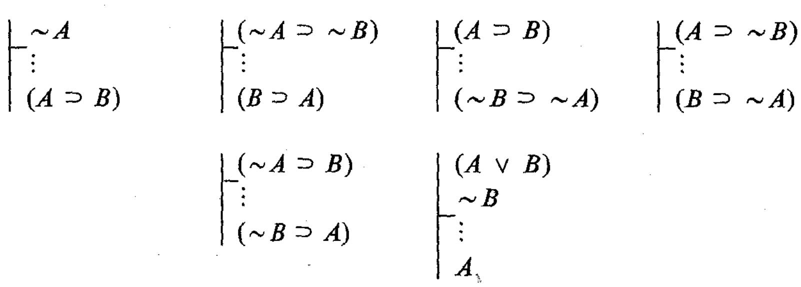
(*xxxi*)



(*xxxii*)

Notice that contradiction elimination is used in constructing example xxxii.

Below are listed some other argument patterns that are especially useful. Rather than filling in the steps of these patterns, however, we will simply list their premisses and conclusions. Their completion is left to you.



$\sim(A \vee B)$ ⋮ $(\sim A \wedge \sim B)$	$(\sim A \wedge \sim B)$ ⋮ $\sim(A \vee B)$	$\sim(A \wedge B)$ ⋮ $(\sim A \vee \sim B)$	$(\sim A \vee \sim B)$ ⋮ $\sim(A \wedge B)$
$(\sim A \vee B)$ ⋮ $(A \supset B)$	$(A \supset B)$ ⋮ $(\sim A \vee B)$	$\sim(A \supset B)$ ⋮ $(A \wedge \sim B)$	$(A \wedge \sim B)$ ⋮ $\sim(A \supset B)$

A final warning. The instructions above will usually result in successful discovery of a derivation, if there is one to be found. Practice in using them is all that is required. But if for some reason you set out to derive something that is underivable in S_s, they of course will not generate an answer; these are instructions for finding a derivation in case one exists, not for showing that there is no derivation in case none exists. One way students sometimes get into trouble is by setting out to derive something that is underivable; then, in order to get an answer, they misapply the rules of S_s. (Deriving '$\sim$p' from '$\sim$(p $\wedge$ q)' by "conj elim" is a typical mistake of this sort.) So, if you find yourself unable to find a correct derivation of one of the exercises, always check to see whether you have copied or translated the exercise properly.

Exercises

1. Whenever possible, translate the following English sentences into S_s. If you feel that a successful translation is not possible, explain why. In giving a translation, be sure to specify which English sentence each of the sentence parameters in the translation stands for.
   (a) He'll come tomorrow if his car doesn't break down on the way.
   (b) Both Baltimore and Hagerstown are in Maryland.
   (c) I never learned to speak German well, but I can understand you if you speak slowly.
   (d) Where is the nearest post office?
   (e) The accused is guilty if and only if he was not at home on the night of July 15; but his wife will not testify against him and there are no other witnesses.
   (f) I've only been in town a week, and already I've been to three parties and a concert.
   (g) Everyone in this room is unable to read music.
   (h) You are allowed to walk your dog only if you have him on a leash, and the dog must have a license.
   (i) $2^{10} + 1$ is a prime number just in case it is not divisible by 17 or by 21.
   (j) If the boss is neither at home nor at work, then he is at the golf course or something extraordinary has happened and he has been detained; but if he has been detained we'll know about it soon.

(k) New York is nearer to Cleveland than to Kingston.
(l) If we're only two days behind schedule, we may be able to catch up if we get some luck.
(m) Go five miles down the road and turn left at the fire station.
(n) The best way to get there is to go five miles down the road and turn left at the fire station.
(o) He didn't know that Mozart and his father were musicians.
(p) If the train isn't late, I'll stop at a bar and get a drink; and that's for sure.

2. Translate the following into S_s, again specifying which English sentence each of the sentence variables signifies. Then find a derivation in S_s of the conclusion from the hypotheses.

(a) If $2 < 3$ then not $3 < 2$.
Therefore if $3 < 2$ then not $2 < 3$.
(b) Oscar is at home, or, if not, he left a message.
Therefore if he didn't leave a message, Oscar is at home.
(c) Albert is either a fool or a liar.
If he is a liar, then what he told me about his sister is false, and I'll look like a fool.
Therefore Albert is a fool or I'll look like a fool.
(d) Fort Wayne is neither in Ohio nor in Illinois.
If Fort Wayne is in Cook County, it is in Illinois.
Therefore it is not the case that Fort Wayne is in Cook County and in Indiana.
(e) If John is arrested he will plead guilty and have to pay a large fine, or plead not guilty and go to a lot of trouble.
If he has to pay a large fine, John will go to a lot of trouble.
Therefore John will go to a lot of trouble if he is arrested.
(f) It is not the case that if God exists, there is unnecessary evil in the world.
Therefore God exists and there is not unnecessary evil in the world.

3. Give derivations in S_s of the following, supplying reasons for all steps.

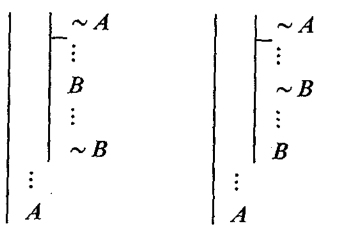
(a) $(q \vee p)$ from $(p \vee q)$
(b) $(q \wedge p)$ from $(p \wedge q)$
(c) $((p \wedge r) \vee q)$ from $((p \vee q) \wedge r)$
(d) $((p \wedge q) \supset r)$ from $(p \supset (q \supset r))$
(e) $(p \supset (q \supset r))$ from $((p \wedge q) \supset r)$
(f) $\sim\sim r$ from r
(g) $(p \supset (q \vee r))$ from $\sim p$
(h) $((p \vee q) \wedge (p \vee \sim q))$ from p
(i) $\sim p$ from $(p \supset \sim p)$
(j) p from $(\sim p \supset p)$
(k) $((p \wedge \sim\sim q) \supset r)$ from $((p \wedge q) \supset r)$
(l) $((p \vee q) \supset r)$ from $((p \vee \sim\sim q) \supset r)$

(m) $(\sim s \supset (r \vee p))$ from $((p \supset q) \supset (r \vee s))$
(n) p from $((p \supset q) \supset q)$ and $(q \supset \sim q)$
(o) $(((q \supset p) \supset p) \supset p)$ from $((p \supset q) \supset p)$
(p) $(r \supset p)$ from $((p \supset q) \supset (r \supset q))$ and $(q \supset p)$
(q) $((p \supset r) \supset r)$ from $((p \supset q) \supset q)$ and $(q \supset r)$
(r) $(((r \supset q) \supset s) \supset ((r \supset p) \supset s))$ from $(p \supset q)$
(s) $(p \vee r)$ from $(p \vee (q \equiv r))$ and $(p \vee q)$
(t) $((p \wedge q) \vee (\sim p \wedge \sim q))$ from $(p \equiv q)$
(u) $(p \vee q)$ from $\sim(p \equiv q)$
(v) $((p \equiv q) \equiv r)$ from $(p \equiv (q \equiv r))$

4. Find categorical derivations in $\mathbf{S_s}$ of the following, supplying a reason for each step.
(a) $(p \vee (p \supset q))$
(b) $((p \supset q) \equiv (\sim p \vee q))$
(c) $((p \vee q) \equiv \sim(\sim p \wedge \sim q))$
(d) $((p \wedge q) \equiv \sim(\sim p \vee \sim q))$
(e) $((((p \wedge q) \vee (\sim p \wedge q)) \vee (p \wedge \sim q)) \vee (\sim p \wedge \sim q))$
(f) $(((p \supset q) \supset p) \supset p)$
(g) $((p \equiv q) \equiv (\sim p \equiv \sim q))$
(h) $(((p \supset q) \supset q) \equiv (p \vee q))$
(i) $((p \vee p) \supset p)$

Problems

1. Devise introduction and elimination rules for the exclusive sense of disjunction, in which the case in which both disjuncts are true is ruled out.
2. Equivalence might have been *defined* in $\mathbf{S_s}$; i.e., formulas such as '$((p \supset q) \wedge (q \supset p))$' might have been used in place of formulas such as '$(p \equiv q)$'. Are there other connectives of $\mathbf{S_s}$ that can be defined in terms of the remaining connectives? How many connectives of $\mathbf{S_s}$ can be eliminated in this way? What are the criteria that allow us to say that a connective may be defined in terms of others?
3. Let the system $\mathbf{S_s}'$ be like $\mathbf{S_s}$ except that the rules of neg int and neg elim are replaced by the following rule, discussed in Section 11.



Show that $\mathbf{S_s}$ and $\mathbf{S_s}'$ are equivalent systems. (Clearly, $\mathbf{S_s}$ and $\mathbf{S_s}'$ are not *identical* systems, since they have different rules of inference, so different arrays of formulas will count as proofs in them. Part of the problem is to figure out an appropriate sense of "equivalence" in which the two systems *are* equivalent.)

```
 1  | |_((p ⊃ (q ⊃ r)) ⊃ s)                 hyp
 2  | | |_r                                 hyp
 3  | | | |_p                               hyp
 4  | | | | |_q                             hyp
 5  | | | | | r                             2, reit
 6  | | | | (q ⊃ r)                         4–5, imp int
 7  | | | (p ⊃ (q ⊃ r))                     3–6, imp int
 8  | | | ((p ⊃ (q ⊃ r)) ⊃ s)               1, reit
 9  | | | s                                 7, 8, m p
10  | | (r ⊃ s)                             2–9, imp int
11  | (((p ⊃ (q ⊃ r)) ⊃ s) ⊃ (r ⊃ s))       1–10, imp int
```

(e)

```
 1  | |_((p ⊃ p) ⊃ q)                       hyp
 2  | | |_p                                 hyp
    | | |
 3  | | (p ⊃ p)                             2, imp int
 4  | | q                                   1, 3, m p
 5  | (((p ⊃ p) ⊃ q) ⊃ q)                   1–4, imp int
```

(f)

```
 1  | |_(p ⊃ q)                             hyp
 2  | | |_((r ⊃ q) ⊃ s)                     hyp
 3  | | | |_(r ⊃ p)                         hyp
 4  | | | | |_r                             hyp
 5  | | | | | (r ⊃ p)                       3, reit
 6  | | | | | p                             4, 5, m p
 7  | | | | | (p ⊃ q)                       1, reit
 8  | | | | | q                             6, 7, m p
 9  | | | | (r ⊃ q)                         4–8, imp int
10  | | | | ((r ⊃ q) ⊃ s)                   2, reit
11  | | | | s                               9, 10, m p
12  | | | ((r ⊃ p) ⊃ s)                     3–11, imp int
13  | | (((r ⊃ q) ⊃ s) ⊃ ((r ⊃ p) ⊃ s))     2–12, imp int
14  | ((p ⊃ q) ⊃ (((r ⊃ q) ⊃ s) ⊃ ((r ⊃ p) ⊃ s)))
                                          1–13, imp int
```

(g)

Chapter IV

1. (a) '(~p ⊃ q)', where 'p' stands for 'His car breaks down on the way' and 'q' for 'He'll come tomorrow'.

   (b) '(p ∧ q)', where 'p' stands for 'Baltimore is in Maryland' and 'q' for 'Hagerstown is in Maryland'.

   (c) '(p ∧ (q ⊃ r))', where 'p' stands for 'I never learned to speak German well', 'q' for 'You speak slowly', and 'r' for 'I can understand you'.

(d) Untranslatable, since it is a question.

(e) '((p ≡ ∼q) ∧ (∼r ∧ ∼s))', where 'p' stands for 'The accused is guilty', 'q' for 'He was at home on the night of July 15', 'r' for 'His wife will testify against him', and 's' for 'There are other witnesses'.

(f) '(p ∧ (q ∧ r))', where 'p' stands for 'I've only been in town a week', 'q' for 'I've already been to five parties', and 'r' for 'I've already been to a concert'.

(g) 'p', where 'p' stands for 'Everyone in this room is unable to read music'.

(h) '((p ⊃ q) ∧ r)', where 'p' stands for 'You are allowed to walk your dog', 'q' for 'You have him on a leash', and 'r' for 'The dog must have a license'. '((p ⊃ q) ∧ (p ⊃ r))' is also an acceptable translation, but results from a less likely interpretation of the sentence.

(i) '(p ≡ (∼q ∧ ∼r))', where 'p' stands for '$2^{10} + 1$ is a prime number', 'q' for '$2^{10} + 1$ is divisible by 17' and 'r' for '$2^{10} + 1$ is divisible by 21'.

(j) '((∼(p ∨ q) ⊃ (r ∨ (s ∧ t))) ∧ (t ⊃ u))', where 'p' stands for 'The boss is at home', 'q' for 'The boss is at work', 'r' for 'He is at the golf course', 's' for 'Something extraordinary has happened', 't' for 'He has been detained', and 'u' for 'We'll know about it soon'.

(k) 'p', where 'p' stands for 'New York is nearer to Cleveland than to Kingston'.

(l) '(q ⊃ (p ⊃ r))', where 'p' stands for 'We get some luck', 'q' for 'We're only two days behind schedule', and 'r' for 'We may be able to catch up'.

(m) Untranslatable, since the sentence is an imperative.

(n) 'p', where 'p' stands for 'The best way to go there is to go five miles down the road and turn left at the fire station'.

(o) '∼p', where 'p' stands for 'He knew that Mozart and his father were musicians'.

(p) '((∼p ⊃ (q ∧ r)) ∧ s)' where 'p' stands for 'The train is late', 'q' for 'I'll stop at a bar', 'r' for 'I'll get a drink', and 's' for 'That's for sure'.

2. (c) From '(p ∨ q)' and '(q ⊃ (r ∧ s))' to derive '(p ∨ s)', where 'p' stands for 'Albert is a fool', 'q' for 'Albert is a liar', 'r' for 'What he told me about my sister is false', and 's' for 'I'll look like a fool'.

1	(p ∨ q)	hyp
2	(q ⊃ (r ∧ s))	hyp
3	p	hyp
4	(p ∨ s)	3, dis int
5	q	hyp
6	(q ⊃ (r ∧ s))	2, reit
7	(r ∧ s)	5, 6, m p
8	s	7, conj elim
9	(p ∨ s)	8, dis int
10	(p ∨ s)	1, 3–4, 5–9, dis elim

(e) From '(p ⊃ ((q ∧ r) ∨ (s ∧ t)))' and '(r ⊃ t)' to derive '(p ⊃ t)', where 'p' stands for 'John is arrested', 'q' for 'He will plead guilty', 'r' for 'He will have to pay a large fine', 's' for 'He will plead not guilty', and 't' for 'He will go to a lot of trouble'.

```
 1   | (p ⊃ ((q ∧ r) ∨ (s ∧ t)))        hyp
 2   |_(r ⊃ t)                          hyp
 3   |   |_p                            hyp
 4   |   | (p ⊃ ((q ∧ r) ∨ (s ∧ t)))    1, reit
 5   |   | ((q ∧ r) ∨ (s ∧ t))          3, 4, m p
 6   |   |   |_(q ∧ r)                  hyp
 7   |   |   | r                        6, conj elim
 8   |   |   | (r ⊃ t)                  2, reit
 9   |   |   | t                        7, 8, m p

10   |   |   |_(s ∧ t)                  hyp
11   |   |   | t                        10, conj elim
12   |   | t                            5, 6–9, 10–11, dis elim
13   | (p ⊃ t)                          1–12, imp int
```

(f) From '~(p ⊃ q)' to infer '(p ∧ ~q)', where 'p' stands for 'God exists' and 'q' for 'There is unnecessary evil in the world'.

```
 1   |_~(p ⊃ q)                  hyp
 2   |   |_~p                    hyp
 3   |   |   |_p                 hyp
 4   |   |   |   |_~q            hyp
 5   |   |   |   | p             3, reit
 6   |   |   |   | ~p            2, reit
 7   |   |   | ~~q               4–6, neg int
 8   |   |   | q                 7, neg elim
 9   |   | (p ⊃ q)               3–8, imp int
10   |   | ~(p ⊃ q)              1, reit
11   | ~~p                       2–10, neg int
12   | p                         11, neg elim
13   |   |_q                     hyp
14   |   |   |_p                 hyp
15   |   |   | q                 13, reit
16   |   | (p ⊃ q)               14–15, imp int
17   |   | ~(p ⊃ q)              1, reit
18   | ~q                        13–17, neg int
19   | (p ∧ ~q)                  12, 18, conj int
```

3.

```
 1   |_(p ∨ q)             hyp
 2   |   |_p               hyp
 3   |   | (q ∨ p)         2, dis int

 4   |   |_q               hyp
 5   |   | (q ∨ p)         4, dis int
 6   | (q ∨ p)             1, 2–3, 4–5, dis elim
```

(a)

```
 1  |_p                            hyp
 2  | (p ∨ q)                      1, dis int
 3  | (p ∨ ~q)                     1, dis int
 4  | ((p ∨ q) ∧ (p ∨ ~q))         2, 3, conj int
```

(h)

```
 1  | ((p ⊃ q) ⊃ q)                hyp
 2  |_(q ⊃ ~q)                     hyp
 3  |  |_~p                        hyp
 4  |  |  |_p                      hyp
 5  |  |  |  |_~q                  hyp
 6  |  |  |  |  p                  4, reit
 7  |  |  |  |  ~p                 3, reit
 8  |  |  |  ~~q                   5–7, neg int
 9  |  |  |  q                     8, neg elim
10  |  |  (p ⊃ q)                  4–9, imp int
11  |  |  ((p ⊃ q) ⊃ q)            1, reit
12  |  |  q                        10, 11, m p
13  |  |  (q ⊃ ~q)                 2, reit
14  |  |  ~q                       12, 13, m p
15  |  ~~p                         3–14, neg int
16  |  p                           15, neg elim
```

(n)

```
 1  | ((p ⊃ q) ⊃ (r ⊃ q))                  hyp
 2  |_(q ⊃ p)                              hyp
 3  |  |_r                                 hyp
 4  |  |  |_~p                             hyp
 5  |  |  |  |_p                           hyp
 6  |  |  |  |  |_~q                       hyp
 7  |  |  |  |  |  p                       5, reit
 8  |  |  |  |  |  ~p                      4, reit
 9  |  |  |  |  ~~q                        6–8, neg int
10  |  |  |  |  q                          9, neg elim
11  |  |  |  (p ⊃ q)                       5–10, imp int
12  |  |  |  ((p ⊃ q) ⊃ (r ⊃ q))           1, reit
13  |  |  |  (r ⊃ q)                       11, 12, m p
14  |  |  |  r                             3, reit
15  |  |  |  q                             13, 14, m p
16  |  |  |  (q ⊃ p)                       2, reit
17  |  |  |  p                             15, 16, m p
18  |  |  ~~p                              4–17, neg int
19  |  |  p                                18, neg elim
20  |  (r ⊃ p)                             3–19, imp int
```

(p)

```
 1  |_~(p ≡ q)                         hyp
 2  |  |_~(p ∨ q)                      hyp
 3  |  |  |_p                          hyp
 4  |  |  |  |_~q                      hyp
 5  |  |  |  | p                       3, reit
 6  |  |  |  | (p ∨ q)                 5, dis int
 7  |  |  |  | ~(p ∨ q)                2, reit
 8  |  |  |  ~~q                       4–7, neg int
 9  |  |  | q                          8, neg elim

10  |  |  |_q                          hyp
11  |  |  |  |_~p                      hyp
12  |  |  |  | q                       10, reit
13  |  |  |  | (p ∨ q)                 12, dis int
14  |  |  |  | ~(p ∨ q)                2, reit
15  |  |  |  ~~p                       11–14, neg int
16  |  |  | p                          15, neg elim
17  |  |  (p ≡ q)                      3–9, 10–16, eqv int
18  |  |  ~(p ≡ q)                     1, reit
19  |  ~~(p ∨ q)                       2–18, neg int
20  |  (p ∨ q)                         19, neg elim
```

(u)

4.

```
 1  |  |_~(p ∨ (p ⊃ q))                hyp
 2  |  |  |_p                          hyp
 3  |  |  |  |_~q                      hyp
 4  |  |  |  | p                       2, reit
 5  |  |  |  | (p ∨ (p ⊃ q))           4, dis int
 6  |  |  |  | ~(p ∨ (p ⊃ q))          1, reit
 7  |  |  |  ~~q                       3–6, neg int
 8  |  |  | q                          7, neg elim
 9  |  |  (p ⊃ q)                      2–8, imp int
10  |  |  (p ∨ (p ⊃ q))                9, dis int
11  |  ~~(p ∨ (p ⊃ q))                 1–10, neg int
12  |  (p ∨ (p ⊃ q))                   11, neg elim
```

(a)

1	$(p \supset q)$	hyp
2	$\sim(\sim p \vee q)$	hyp
3	p	hyp
4	$(p \supset q)$	1, reit
5	q	3, 4, m p
6	$(\sim p \vee q)$	5, dis int
7	$\sim(\sim p \vee q)$	2, reit
8	$\sim p$	3–7, neg int
9	$(\sim p \vee q)$	8, dis int
10	$\sim\sim(\sim p \vee q)$	2–9, neg int
11	$(\sim p \vee q)$	10, neg elim
12	$(\sim p \vee q)$	hyp
13	$\sim p$	hyp
14	p	hyp
15	$\sim q$	hyp
16	p	14, reit
17	$\sim p$	13, reit
18	$\sim\sim q$	15–17, neg int
19	q	18, neg elim
20	$(p \supset q)$	14–19, imp int
21	q	hyp
22	p	hyp
23	q	21, reit
24	$(p \supset q)$	22–23, imp int
25	$(p \supset q)$	12, 13–20, 21–24, dis elim
26	$((p \supset q) \equiv (\sim p \vee q))$	1–11, 12–25, eqv int

(b)

1	$((p \supset q) \supset p)$	hyp
2	$\sim p$	hyp
3	p	hyp
4	$\sim q$	hyp
5	p	3, reit
6	$\sim p$	2, reit
7	$\sim\sim q$	4–6, neg int
8	q	7, neg elim
9	$(p \supset q)$	3–8, imp int
10	$((p \supset q) \supset p)$	1, reit
11	p	9, 10, m p
12	$\sim\sim p$	2–11, neg int
13	p	12, neg elim
14	$(((p \supset q) \supset p) \supset p)$	1–13, imp int

(f)