[11-02-03-Th]

It is high time now to put these ideas to work within the context of some formal system. For this purpose we'll work at first with a class of formulas that in a way is more general, and in a way more restricted than the class of formulas of S_0. We will allow more (infinitely more!) sentence parameters than the four of S_0 but will not, for the time being, allow the use of negation.

The formulas of $S_\supset$, then, are defined as follows.

1. Any of the sentence parameters

$$\begin{array}{lllll} p & q & r & s & \ldots \\ p_1 & q_1 & r_1 & s_1 & \ldots \\ p_2 & q_2 & r_2 & s_2 & \ldots \\ & \vdots & & & \end{array}$$

   is a formula of $S_\supset$;

2. If A and B are formulas of $S_\supset$, then so is $(A \supset B)$.

A string of symbols qualifies as a formula of $S_\supset$ only if it can be shown to be a formula by repeated applications of 1 and 2.

Derivations in $S_\supset$ will use the format we developed in the previous section, the steps in these derivations being formulas rather than English sentences. Knowing only this much, we can form the following derivations in $S_\supset$.

```
|_p
|   |_(q ⊃ r)
|   | p
      (ix)
```

```
|_(p ⊃ q)
|   |_p
|   |   |_(q ⊃ r)
|   |   | p
|   |   | (p ⊃ q)
            (x)
```

But we aren't going to get very far this way, just making hypotheses and using the reiteration rule. We need additional rules in order to really do something with the system. We don't have to look very far for such rules; we already used one likely candidate in examples iii and viii. This rule, you remember, allowed us to infer an implication (like '(p ⊃ q)') from a derivation of its consequent ('q', in this case) from its antecedent ('p'). Schematically, we can picture the rule as follows.

```
|   |_A
|   | ⋮
|   | B
| (A ⊃ B)
```

This diagram means that whenever in a derivation we have completed a pattern looking like

```
| |_A
| | :
| | B
```

(so that 'A' and 'B' will be replaced by particular formulas, and the dots by an argument using rules of $S_\supset$), we can go on to infer $(A \supset B)$ as shown.

We will call this rule the rule of *implication introduction*, since it allows us to infer *to* an implication; it shows us how to introduce implications in an argument by means of reasoning.

The rule of implication introduction can be combined with the reiteration rule to produce more complicated and interesting derivations, such as these.

```
|_(p ⊃ q)
|  |_p
|  | (p ⊃ q)
| (p ⊃ (p ⊃ q))
     (xi)
```

```
| |_p
| |  |_q
| |  | p
| | (q ⊃ p)
| (p ⊃ (q ⊃ p))
     (xii)
```

We can also use the rule of implication introduction to derive formulas such as '$(p \supset p)$' categorically; this can be done in the following way.

```
| |_p
| |
| (p ⊃ p)
   (xiii)
```

In a way, this is a lazy application of implication introduction, since nothing at all is derived from the hypothesis 'p' before it is discharged; but nevertheless, we will regard examples such as this as correct uses of implication introduction.

6. There is just one more thing that needs to be added in order to complete the system $S_\supset$. There are various ways to see what is needed; one way is to try to see what would be required to derive '$((p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r)))$' categorically.

But general considerations of symmetry also suggest that $S_\supset$ should have a rule of *implication elimination* to complement its rule of implication introduction. After all, we need to argue *from* implications as well as *to* them.

Again there is little doubt about what this rule ought to be; the rule of *modus ponens* is the most plausible candidate. Putting this suggestion into practice, let's agree that applications of the rule of implication elimination (or *modus ponens*) are permitted in the system $S_{\supset}$ as follows.

```
| A              | A ⊃ B
| ⋮              | ⋮
| A ⊃ B          | A
| ⋮              | ⋮
| B              | B
```

That is, any time when we have both formulas A and $A \supset B$ in a derivation, we may infer B by the rule of implication elimination.

In the following examples all the rules of $S_{\supset}$ are employed.

```
| |_(p ⊃ (q ⊃ r))
| | |_(p ⊃ q)
| | | |_p
| | | | (p ⊃ q)
| | | | q
| | | | (p ⊃ (q ⊃ r))
| | | | (q ⊃ r)
| | | | r
| | | (p ⊃ r)
| | ((p ⊃ q) ⊃ (p ⊃ r))
| ((p ⊃ (q ⊃ r)) ⊃ ((p ⊃ q) ⊃ (p ⊃ r)))
```

(*xiv*)

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

(*xv*)

As in the case of S_0, it often is helpful to annotate derivations, showing the reasons for steps. This will become particularly useful later on when we will be working with more rules. Example xv, if annotated in this way, looks like this.

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

(*xv′*)

7. For some purposes it is convenient to have derivations with multiple hypotheses. (One reason for this is that in everyday cases we frequently want to argue from many hypotheses—in no particular order of subordination—down to a conclusion.) It is easy to allow such derivations in $S_\supset$. Here is an example, closely related to example xiv above.

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

(*xvi*)

8. There are several mistakes that beginners often make in trying to find derivations in systems such as $S_\supset$. One of the most common misunderstandings leading to mistakes of this kind has to do with subordination.

For instance, suppose we are set the task of deriving 'r' from 'q' and '((p ⊃ q) ⊃ r)'. In the course of solving this problem we may be forced to make hypotheses; and since we can make any hypothesis we like, why not assume '(p ⊃ q)'? This will enable us to get to 'r' as follows.

```
1   | q                     hyp
2   |_((p ⊃ q) ⊃ r)         hyp
3   |  |_(p ⊃ q)            hyp
4   |  |  ((p ⊃ q) ⊃ r)     2, reit
5   |  |  r                 3, 4, m p
```

(*xvii*)

The trouble with xvii is that it isn't a solution to the problem. Its last step, 'r', is indeed the conclusion we wanted; but in this step 'r' has not been obtained as an item of the main derivation; it is an item of a subordinate derivation in which a further hypothesis is made. In step 5, 'r' is subject not only to the hypotheses 'q' and '((p ⊃ q) ⊃ r)', but also to the hypothesis '(p ⊃ q)'.

To see just how mistaken this technique is, it's only necessary to consider an extreme example of it like the following "derivation" of 'q' from 'p'.

```
|_p
|  |_q
|  |
```

(*xviii*)

This example is the most blatant possible case of the fallacy known as *petitio principii*: assuming what was to be proved. Example xvii is a more disguised version of the same fallacy. Using this method, it would be easy to "derive" anything from anything.

What is asked, then, by a request to derive B from hypotheses $A_1, \ldots, A_n$ is that B should be obtained as an item in a derivation headed by the hypotheses $A_1, \ldots, A_n$—not as an item in a derivation in which other hypotheses are in force as well. To derive B from $A_1, \ldots, A_n$ means to derive B from $A_1, \ldots, A_n$ *only*.

On the other hand, this doesn't mean that other hypotheses can't be made in carrying out a derivation. It means only that these hypotheses must be *discharged* in the course of the argument.

Consider, for instance, our original problem: to derive 'r' from 'q' and '((p ⊃ q) ⊃ r)'. The insight needed here is that we could obtain 'r' by *modus ponens* if we could get '(p ⊃ q)'. We are then led to attempt a derivation along the following lines.

```
| q
|_((p ⊃ q) ⊃ r)
| ?
| (p ⊃ q)
| r
```

(*xix*)

In a situation like this, when you're trying to argue to an implication, it is never a bad idea to try to get it by implication introduction. So we set up a subordinate derivation with 'p' as hypothesis, and try to obtain 'q' in this new derivation.

```
| q
|_((p ⊃ q) ⊃ r)
|  |_p
|  |  ?
|  |  q
| (p ⊃ q)
| r
```

(*xx*)

The last step of this subordinate derivation isn't an implication, so we can't get it by implication introduction. Therefore we look up at the formulas that can be reiterated into the derivation, to see whether we can use them to get 'q'. In this case, it's easy; 'q' *is* one of these formulas, and we can end the derivation as it stands. The end product looks like this.

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

(*xxi*)

Notice that although another hypothesis has been made in xxi, it has been discharged by step 6. In this step 'r' is subject to only the hypotheses 'q' and '((p ⊃ q) ⊃ r)', as is shown by its not being separated by any lines from the outermost line of the derivation.

Another misapprehension that sometimes causes trouble is that hypotheses made in a derivation must be *used* in getting to the conclusion. So far is this from being true that very often we have to make hypotheses that are quite irrelevant to the conclusion. This occurs, for instance, in trying to find a categorical derivation of '(p ⊃ (q ⊃ q))'.

In solving this problem, we should follow the policy of getting implications by implication introduction and begin as follows.

```
|  |_p
|  |  ?
|  |  (q ⊃ q)
| (p ⊃ (q ⊃ q))
```

(*xxii*)

The danger here is to assume that '(q ⊃ q)' must come by somehow using the hypothesis 'p'. If you try to do this you'll get stuck.

The right way to proceed in cases such as this is to rely mechanically on the advice that implications should be obtained by implication introduction; we then form a subordinate derivation with 'q' as hypothesis.

```
| |_p
| |  |_q
| |  |  ?
| | (q ⊃ q)
| (p ⊃ (q ⊃ q))
```

(xxiii)

Since, as we explained in Section 5, we regard steps such as

```
| |_q
| |
| (q ⊃ q)
```

as instances of implication introduction, we need proceed no further; to finish the problem it's only necessary to fill in reasons.

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

(xxiv)

This is a perfectly correct derivation, even though no use whatsoever was made of the hypothesis 'p' in deriving '(q ⊃ q)'.

Exercises

1. Find derivations of the following in $S_{\supset}$.

   (a) (p ⊃ q) from (r ⊃ q) and (p ⊃ r)
   (b) (r ⊃ (p ⊃ (q ⊃ r))) from (p ⊃ (r ⊃ (q ⊃ r)))
   (c) (p ⊃ (p ⊃ p)) from q
   (d) (p ⊃ p) from (q ⊃ r)
   (e) r from ((p ⊃ (q ⊃ p)) ⊃ r)
   (f) ((r ⊃ q) ⊃ q) from ((p ⊃ q) ⊃ q) and (p ⊃ r)
   (g) ((p ⊃ r) ⊃ r) from ((p ⊃ q) ⊃ q), (q ⊃ r), and (r ⊃ q)
   (h) r from p and (((q ⊃ q) ⊃ p) ⊃ r)

(i) ((p ⊃ q) ⊃ (r ⊃ r)) from ((p ⊃ q) ⊃ (r ⊃ q))
(j) p from ((p ⊃ q) ⊃ p), (p ⊃ (r ⊃ q)), and r
(k) (r ⊃ p) from ((p ⊃ r) ⊃ ((q ⊃ r) ⊃ (r ⊃ p)))
(l) (p ⊃ q) from (((r ⊃ q) ⊃ (q ⊃ p)) ⊃ (p ⊃ q))

2. Find categorical derivations of the following in $S_\supset$.
(a) ((p ⊃ q) ⊃ ((q ⊃ r) ⊃ (p ⊃ r)))
(b) (r ⊃ ((p ⊃ q) ⊃ (s ⊃ (p ⊃ q))))
(c) (((p ⊃ p) ⊃ p) ⊃ p)
(d) ((p ⊃ (p ⊃ q)) ⊃ (p ⊃ q))
(e) (((p ⊃ (q ⊃ r)) ⊃ s) ⊃ (r ⊃ s))
(f) (((p ⊃ p) ⊃ q) ⊃ q)
(g) ((p ⊃ q) ⊃ (((r ⊃ q) ⊃ s) ⊃ ((r ⊃ p) ⊃ s)))

Problems

1. Show that if there is a categorical derivation of a formula *A* in $S_\supset$ then there is a categorical derivation of (*B* ⊃ *A*) in $S_\supset$, where *B* is any formula of $S_\supset$.
2. Show that, if there is a categorical derivation of a formula *A* in $S_\supset$, then there is a categorical derivation of any substitution instance of *A*.
3. Is there a categorical derivation in $S_\supset$ of '(((p ⊃ q)⊃ p)⊃ p)'? If you claim there is none, can you devise any clearcut way of demonstrating that your claim is true?
4. In the next chapter, we are going to add rules for negation ('∼'), conjunction ('∧'), disjunction ('∨'), and equivalence ('≡'). Figure out what these rules must be.

13	$(((s \supset r) \supset (q \supset (s \supset r))) \supset ((s \supset r) \supset ((q \supset s) \supset (q \supset r))))$	7, 12 m p 4, subst
14	$((s \supset r) \supset (q \supset (s \supset r)))$	13, 14, m p
15	$((s \supset r) \supset ((q \supset s) \supset (q \supset r)))$	15, subst
16	$((p \supset r) \supset ((q \supset p) \supset (q \supset r)))$	16, subst
17	$((p \supset s) \supset ((q \supset p) \supset (q \supset s)))$	17, subst
18	$((p \supset s) \supset ((r \supset p) \supset (r \supset s)))$	18, subst
19	$((p \supset q) \supset ((r \supset p) \supset (r \supset q)))$	

Chapter II

1. (a) '$(\sim q \supset p)$', where 'p' stands for 'We'll go to the beach today' and 'q' for 'It rains'.
   (b) '$(p \supset \sim q)$', where 'p' stands for 'You do what you're told' and 'q' for 'You'll get along badly here'.
   (c) '$\sim q$', where 'q' stands for 'The dog was treated unkindly'.
   (d) '$(p \supset (q \supset (\sim r \supset s)))$', where 'p' stands for 'I move my pawn', 'q' for 'He castles', 'r' for 'I lose my queen', and 's' for 'I should be able to beat him'.
   (e) 'p', where 'p' stands for 'All mice are mortal'.
   (f) '$(p \supset (q \supset r))$', where 'p' stands for 'The next train is on time', 'q' for 'I miss my train', and 's' for 'I can arrive only five minutes late'.
   (g) 'p', where 'p' stands for 'Sam isn't over five feet tall, unless he has grown'.

Chapter III

1. (c)

1	| q	hyp
2	| | p	hyp
3	| | | p	hyp
4	| | $(p \supset p)$	3, imp int
5	| $(p \supset (p \supset p))$	2–4, imp int

(e)

1	| $((p \supset (q \supset p)) \supset r)$	hyp
2	| | p	hyp
3	| | | q	hyp
4	| | | p	2, reit
5	| | $(q \supset p)$	3–4, imp int
6	| $(p \supset (q \supset p))$	2–5, imp int
7	| r	1, 6, m p

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

(h)

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

(j)

2.

```
 1  |  |_r                                        hyp
 2  |  |  |_(p ⊃ q)                               hyp
 3  |  |  |  |_s                                  hyp
 4  |  |  |  | (p ⊃ q)                            2, reit
 5  |  |  | (s ⊃ (p ⊃ q))                         3–4, imp int
 6  |  | ((p ⊃ q) ⊃ (s ⊃ (p ⊃ q)))                2–5, imp int
 7  | (r ⊃ ((p ⊃ q) ⊃ (s ⊃ (p ⊃ q))))             1–6, imp int
```

(b)

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

(c)

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

(d)

```
 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'.