Introducing *SD*

And with that we have finished with the semantics of our language of *SL*. But there is more syntax left to do!

Earlier, we completely defined the syntax of our language of *SL*—we gave a complete and precise specification of exactly what the symbols and sentences of *SL* were.

But logic is more than a language for saying stuff. It is also intended to provide a model for inference or reasoning. And we want this inference model to be completely syntactic in its specification. This way, our model of good and bad inferences will not depend upon the meaning of the sentences involved in the inference, but will be based purely upon their logical form.

Introducing *SD* Continued:

So we will develop purely formal rules which allow us to go from sentences to sentences in a way intended to mirror inference. The rules will be defined solely in terms of the "shapes" of the sentences involved. They will all be of the form:

> Given squiggles that look like this, you are entitled to write down a squiggle that looks like that.

Our rules will be chosen with the meaning of the logical symbols in mind. But for the application of the rules, the meanings are irrelevant. Only the form or shape of the sentences in question will matter.

Introducing *SD* Continued:

There are two main ways to present a deductive of inferential system: *Axiomatic Deduction* and *Natural Deduction*.

Axiomatic systems generally have very few rules and a number of sentences, called axioms, that you are allowed to write down whenever you want. Much serious research work and logical theory is done with axiomatic systems. For reasons which will become clear later in the course, having few rules makes it relatively easy to prove that your system of inference has various desirable properties.

However, axiomatic systems are also generally relatively difficult to actually use. Constructing proofs or derivations in axiomatic systems is fairly difficult. Additionally, axiomatic reasoning bears little resemblance to the sort of reasoning the people actually engage in.

Introducing *SD* Continued:

In contrast to axiomatic systems, Natural Deduction systems are considerably easier to employ. The cost is that proofs about the Natural Deduction system are often considerably more work.

Both features are because, in a Natural Deduction system, we have relatively many rules as compared to axiomatic systems. (We also have no sentences that we are allowed to write down whenever we want—we have no axioms.)

The various rules of a Natural Deduction system are designed to be fairly close representations of the sort of reasoning that people actually engage in. Hence the name: *Natural* Deduction.

Introducing *SD* Continued:

In this course we will use Natural Deduction presentations. The system of inference that we will develop for inferences on sentences of *SL* is called *SD* for *Sentential Deduction*.

There are other methods of presenting deductive systems as well. Sequent Calculi are very useful and “natural” for certain applications of logic. There is also the *Tree Method* which can arguably be seen as a syntactic deductive system. (A tree method is presented in the chapters of *The Logic Book* which we will be skipping.)

Introducing *SD* Continued:

Our system of *SD* will have eleven rules—for each of our five sentential connectives we will have one rule to introduce that connective and one rule to eliminate that same connective. We will also have a rule which allows us to repeat a previous sentence (subject to some restriction) in order to "move" it to a place where we need it. (This last rule we will call the rule of Reiteration.)

The text also presents an extended Natural Deduction system SD^+. Its additional rules make constructing derivations in SD^+ even easier than constructing them in *SD*. But the cost is that proof that SD^+ has various desirable properties are even more cumbersome than the parallel proofs that *SD* has those properties. Thus, we won't be using SD^+ in this course.

Introducing *SD* Continued:

And this has been a lot of text! But we are almost ready to begin presenting the system *SD*. Only one caveat remains:

SD is a *system*. And thus it can really only be understood when you look at its parts together. Thus, in the next seven or so slides, there will be a good deal of the system that we leave mentioned but not explained. This is because we need to have seen enough of the system for an explanation of the parts we elide over to make sense. But, have no fear, it will all be explained before we are done!

Also, I am going to present the rules in a different order and somewhat differently than the text does. The order is changed to present the easiest rules first, only then moving to the tricky ones. The manner of presentation differs so as to be more precise. But the rules are the very same.

Preamble over, on with the presentation of the system. We start with the easiest rule of all, the rule of Reiteration.

Reiteration (p. 146/162)

The Reiteration rule of *SD* allows us to repeat a previously occuring and accessible line of a derivation. (What "accessibility" means will be explained soon.) It looks like this:

i	P	
k	P	*i* R

Introduction Rule for '&' (p. 147/163)

The '&' Introduction rule allows us to write down a sentence of the form P & Q whenever we have both P and Q available on previous accessible lines of a derivation. It looks like this:

i	P		
j	Q		
k	P & Q	*i*, *j* & I	

Note that lines *i* and *j* can occur in either order; it doesn't matter which.

Elimination Rule for '&' (p. 147/163)

The '&' Elimination rule allows us to write down a sentence of the form P or a sentence of the form Q whenever we have P & Q available on previous accessible lines of a derivation. It looks like this:

i		P & Q	
k		P	*i* & E

or

i		P & Q	
k		Q	*i* & E

So, from a conjunction, we can get either conjunct.

Introduction Rule for '$\vee$' (p. 155/171)

The '$\vee$' Introduction rule allows us to write down a sentence of the form $P \vee Q$ or of the form $Q \vee P$ whenever we have P available on a previous accessible line of a derivation. It looks like this:

i	|	P	
	|		
	|		
k	|	$P \vee Q$	*i* $\vee$ I

or

i	|	P	
	|		
	|		
k	|	$Q \vee P$	*i* $\vee$ I

So, from a sentence, we can get a disjunction where that sentence is either the left disjunct, or the right disjunct.

Elimination Rule for '$\supset$' (p. 149/165)

The '$\supset$' Elimination rule allows us to write down a sentence of the form Q whenever we have both P and $P \supset Q$ available on previous accessible lines of a derivation. It looks like this:

i	P		
j	$P \supset Q$		
k	Q	*i* , *j*	$\supset$E

Note that lines *i* and *j* can occur in either order; it doesn't matter which.

Elimination Rule for '≡' (p. 158/174)

The '≡' Elimination rule allows us to write down a sentence of the form P whenever we have $P \equiv Q$ and Q available on previous accessible lines or a sentence of the form Q whenever we have $P \equiv Q$ and P available on previous accessible lines of a derivation. It looks like this:

i	$P \equiv Q$		
j	P		
k	Q	*i*, *j* ≡ E	

or

i	$P \equiv Q$		
j	Q		
k	P	*i*, *j* ≡ E	

So, from a biconditional and one of its sides, we can get the other side. (Note that lines *i* and *j* can occur in either order.)

Introduction Rule for '$\supset$' (p. 149/165)

The '$\supset$' Introduction rule allows us to write down a sentence of the form $P \supset Q$ whenever we have a subderivation starting in the assumption of P and ending with the sentence Q available on previous accessible lines of a derivation. It looks like this:

i	P	Assumption
j	Q	
k	$P \supset Q$	*i*–*j* $\supset$I

And now you might say:

OK. But what is a subderivation? And while were it, what's up with those lines around the P and Q? And, for that matter, the lines that have been there in all the other rules? And didn't you say you'd explain accessibility at some point?!

Good questions!

Subderivations and Scopelines

In the application of the '$\supset$' Introduction rule to get $P \supset Q$ we make a temporary assumption of P and then go on to get Q. This is what allows us to write down $P \supset Q$:

i	P	Assumption
j	Q	
k	$P \supset Q$	*i*–*j* $\supset$I

But, since we assumed P only in order to get $P \supset Q$, once we have written down $P \supset Q$ we have "used up" the assumption of P—we cannot use it any longer. The lines from P to Q constitute a *subderivation*—they are a derivation that is subsidiary to our main derivation. Once we have finished a subderivation, the assumption of the subderivation—P in our case—is no longer applicable and we say that it has been discharged. And it is no longer accessible.

What does that mean?

Subderivations and Scopelines Continued:

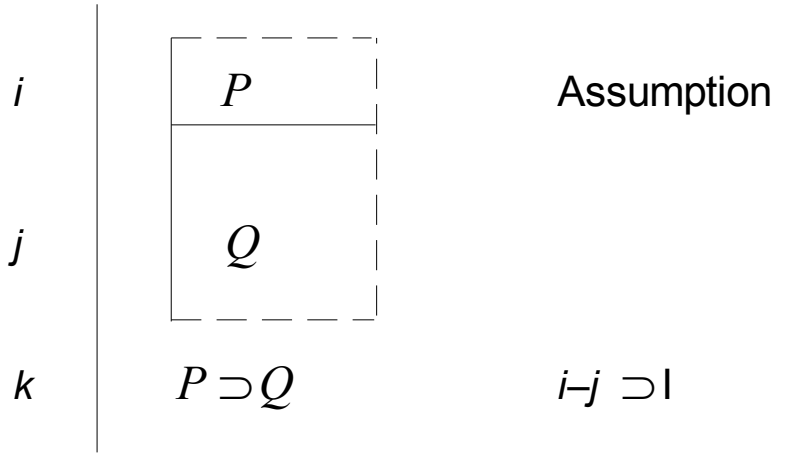
Among other things, that it would be a mistake to go on it the derivation schema above as follows:

i	P	Assumption	
j	Q		
k	$P \supset Q$	*i*–*j* ⊃I	
k + 1	P	*i* R (Reiteration)	✘ **WRONG!**
k + 2	P & $(P \supset Q)$	*i*, *k* & I	✘ **WRONG!**
k + 3	P & Q	*i*, *j* & I	✘ **WRONG!**

At line *k* the assumption has been discharged, so neither it, nor any of the lines of the derivation in its "scope" are available for subsequent rules. They are inaccessible.

Subderivations and Scopelines Continued:

The lines along the sides of various sentences in a derivation are called “scopelines” and they help us track the “scope” or range of availability of the various assumption that we have made. Think of them as abbreviated boxes. As in, mentally add the dotted lines:



Indeed, the original notation for Natural Deduction systems (invented by Gerhard Gentzen in the 1930's) had a solid box around the subderivations. The modern notation is just an abbreviation for that.

Subderivations and Scopelines Continued:

In order to close a subderivation and discharge its assumption, the last line must be immediately to the right of the scopeline of the assumption of the subderivation. Consider the following:

i	P	Assumption	
j	R	Assumption	
k	Q		
k + 1	$P \supset Q$	*i*–*k* $\supset$I	✘ **WRONG!**

This is an error as we have obtained Q—somehow or other—in a way that depends upon our having had both the assumption P and Q available to us. But we have written down something that we are only entitled to do when we have obtained Q from the assumption of P.

Subderivations and Scopelines Continued:

But we can nest subderivations; that is perfectly legal. One way to "repair" the last example to be a perfectly good derivation would be as follows:

i	P	Assumption	
j	R	Assumption	
k	Q		
k + 1	$R \supset Q$	*j*–*k* $\supset$I	✓ **FINE!**
K + 1	$P \supset (R \supset Q)$	*i*–(*k* +1) $\supset$I	✓ **FINE!**

Here we open a subderivation with the assumption of P and then another with the assumption of R and use the resources available to derive Q. We close the assumption of R and write down $R \supset Q$. We then close the assumption of P and write down $P \supset (R \supset Q)$. This closes our first subderivation assumption and we are done.