Introducing Sentential Logic (SL)
Part I – Syntax

1. As Aristotle noted long ago, some entailments in natural language seem to hold by dint of their “logical” form. Consider, for instance, the example of Aristotle’s being a logician above, as opposed to the “material” entailment considering the relative locations of Las Vegas and Pittsburgh. Such logical entailment provides the basic motivation for formal logic. So-called “formal” or “symbolic” logic is meant in part to provide a (perhaps idealized) model of this phenomenon. In formal logic, we develop an artificial but highly regimented language, with an eye perhaps of understanding natural language devices as approximating various operations within that formal language.

2. We’ll begin with a formal language that is commonly called either Sentential Logic (SL) or, more high-falutinly, “the propositional calculus.” You’re probably already quite familiar with it from an earlier logic course.

SL is composed of Roman capital letters (and subscripted capital letters), various operators (or functors), and left and right parentheses. The standard operators for SL include the tilde (~), the ampersand (&), the wedge (v), and the arrow (→). Other operators, such as the double arrow, the stroke and the dagger, can easily be added as the need arises.

The Syntax of SL

3. A syntax for a language specifies how to construct meaningful expressions within that language. For purely formal languages such as SL, we can understand such expressions as well-formed formulas (wffs).

The syntax of SL is recursive. That means that we start with basic (or atomic) wffs, and then specify how to build up more complex wffs from them.

Rules for constructing wffs in SL:

(i) Any capital letter (or subscripted capital letter) is a wff. These are SL’s atomic sentences.

(ii) If some expression Φ is a wff, then so too is the expression ~Φ also a wff. Thus the tilde is called a one-place sentential operator.

(iii) If the expressions Φ and Ψ are both wffs, then so too is (Φ & Ψ), (Φ v Ψ), and (Φ → Ψ). Thus the ampersand, wedge, and arrow are all two-place sentential operators.

(iv) Finally, no other expressions are wffs in SL. This rule tells us not only that SL is recursive, but that it is fully recursive.

4. Our syntax here is unambiguous in the sense that you can take any non-atomic or compound wff and reconstruct exactly how it can be constructed out of simpler components from the rules above. The key
here is that from any compound wff whatsoever, one can identify exactly one of its component sentential operators as its major operator (and then in turn identify the major operator(s) of its component formulas).

(a) A one-place sentential operator (e.g., a tilde) will be a formula’s major operator just in case it prefixes (is placed in front of) the entire rest of the formula (in which case, by rule ii of our syntax, the rest of the formula will itself be a wff).

(b) A two-place sentential operator (ampersand, arrow, wedge) will be a formula’s major operator just in case the entire formula has an outermost set of parentheses, and when those parentheses are removed, one is left with two complete wffs on both sides of that operator.

One can see that the primary role of parentheses in our syntax is that of indicating where the boundaries of component formulas are. For the sake of convenience and ease of reading, one may adopt the convention of omitting a formula’s outermost set of parentheses (bearing in mind that “strictly speaking” they are still there, and that they must be introduced when that formula is embedded inside another!).

5. Some Quick Exercises:

(1) How many distinct ways are there of restoring brackets to form wffs out of the following string?

\[ \sim P \& Q \rightarrow \sim R \lor S \]

(2) Explain why, with our existing syntax, we could or could not get away with just using a single type of punctuation mark (say, a stove pipe, |, that does not indicate whether it is oriented to the right or to the left.

(3) Suppose instead that we were working with a formal language consisting entirely of atomic propositions (symbolized by capital letters), a one-place functor *", and a three-place functor #", both of which “prefix” (or are written in front of) the requisite string of wffs. Write out rules, analogous to (i)-(iv) in section 6. above, for a fully-recursive syntax of this language.

(4) Explain why such a language might not need any parentheses.

(5) Finally, give a line of reasoning showing that in this language (where formulas are constructed entirely out of atomic propositions and a one and a three-place operator) no well-formed formula could ever have an even number of atomic proposition letters.