6.1.s. Summary

We move beyond truth-functional logic by recognizing complete expressions other than sentences and operations other than connectives. Our additions are motivated by a traditional description of grammatical subjects and predicates. The new complete expressions are individual terms, whose function is to name. Given this idea, we can define a predicate as an operation that forms a sentence from one or more individual terms.

A predicate corresponds to an English sentence with blanks that might be filled by terms. These blanks are the predicate's places and the operation of filling them is predication. We will maintain something analogous to truth-functionality by requiring that predicates be extensional. This means that all places of a predicate must be referentially transparent (rather than referentially opaque): when judging the truth value of a sentence formed by the predicate, we must be able see through the terms filling these places to what those terms refer to. Thus, just as a connective expresses a truth function, a predicate expresses a function that takes reference values as input and issues truth values as output. Such a function may be called an attribute—or, more specifically, a property if it has one place and a relation if it has 2 or more. In symbolic notation, it takes the form $\sigma = \tau$ and, in English notation, it takes the form σ `is` τ.

While recognizing quite a variety of non-logical vocabulary in our analyses, we recognize only one new item of logical vocabulary, the predicate identity. This is a 2-place predicate that forms an equation, which is true when its component terms have the same reference value.

Lambda abstraction provides notation for linking the places of a predicate to blanks in an English sentence. An expression formed using it—which will have the general form $\lambda x_1 \ldots x_n (\ldots x_1 \ldots x_n \ldots)$—is an abstract (in this use, a predicate abstract); it consists of a lambda operator applied to a parenthesized body. In English notation, a predicate abstract takes the form `the attribute of` $x_1 \ldots x_n$ `that` $\ldots x_1 \ldots x_n \ldots$. Variables in the body of an abstract are bound to the lambda operator. Expressions that establish the same patterns of binding using different variables are alphabetic variants. They may be thought of as pronouns whose antecedent is the lambda operator. An expression (such as the body of an abstract) that has variables not bound to lambda operators, is not a sentence in the strict sense, but it does count as a formula. Formulas have many of the syntactic properties of sentences; in particular, they can be built from other formulas using connectives. And we can distinguish as atomic formulas not only unanalyzed sentences but all formulas that are predictions. (Indeed, unanalyzed sentences can be thought of as predications of zero-place predicates.)

In our symbolic notation, we use lower case letters to stand for unanalyzed individual terms, the equal sign for identity, and capital letters to stand for non-logical predicates. Non-logical predicates, both capital letters and predicate abstracts are written in front of the terms they apply to (with a predicate abstract enclosed in brackets), and = is written between the terms to which it applies. In English notation, predications other than equations are written as θ `fits` τ_1, ..., `'n` τ_n.