

6. Predications

6.1. Naming and describing

6.1.0. Overview

We will now begin to study a wider variety of logical forms in which we identify components of sentences that are not also sentences.

6.1.1. A richer grammar

A large variety of grammatical categories can be defined once we introduce the idea of an individual term, an expression whose function is to name.

6.1.2. Logical predicates

When the subject is removed from a sentence, a grammatical predicate is left behind; a logical predicate is what is left when any number of individual terms are removed.

6.1.3. Identity

We will study the special logical properties of only one predicate, the one expressed by the equals sign.

6.1.4. Abstracts

A predicate has a number of places in a given order, and abstracts are a notation for associating these places with blanks in a sentence.

6.1.5. Analyzing predications

When the analysis of truth-functional structure is complete, we go on to analyze atomic sentences as predications.

Glen Helman 25 Aug 2005

6.1.1. A richer grammar

While there are more truth-functional connectives that we might study and more questions we might ask about those we have studied, we will now move on from truth-functional logic. The logical forms we will now explore involve ways sentences may be constructed out of expressions that are not yet sentences. Although the kinds of expressions we will identify do not correspond directly to any of the usual parts of speech, our analyses will be comparable in detail to grammatical analyses of short sentences into words.

The simplest case of this sort of analysis is related to, but not identical with, the traditional grammatical analysis into subject and predicate. You might find a grammar text of an old-fashioned sort defining *subject* and *predicate* correlatively as the part of the sentence that is being spoken of and the part that says something about it. Of course, in saying that the subject is being spoken of, there would be no intention to say that the predicate is used to say something about words. So the text might go on to say that a subject contains a word that names the “person, place, thing, or idea” (to quote one of my high school grammar texts) about which something is being said. Thus we have the situation shown in Figure 6.1.1-1.

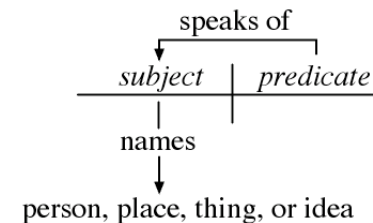


Fig. 6.1.1-1. The traditional picture of grammatical subjects and predicates.

This picture is really not adequate for either grammar or logic, but grammarians and logicians part company in the ways they refine it. Grammarians look for more satisfactory definitions of *subject* and *predicate* that capture, at least roughly, the expressions that have been traditionally labeled in this way. Logicians, on the other hand, accept something like the definitions above and look for expressions that really have the functions they describe, whether or not these expressions would traditionally be labeled subjects and predicates.

“Subjects” and “predicates” in the logical sense provide, along with sentences and connectives, examples of two broad syntactic categories, **complete expressions** and **operations**. Sentences are examples of complete expressions and connectives are examples of operations. Like connectives, operations in general can be thought of as expressions with blanks, expressions that are incomplete in the sense that they are waiting for input. We can classify operations according to the number and kinds of inputs they are waiting for and the kind of output they yield when they receive this input. In the case of connectives, both the input or inputs and the output are sentences.

A “subject” in the logical sense will be a kind of complete expression, an **individual term**. This is a type of expression whose function is to refer to something; that is, it is an expression which can be described, roughly, as naming a “person, place, thing, or idea.” In [6.2.1](#), we will consider the full range of expressions that count as individual terms but, for now, it will be enough to have in mind some basic examples, proper names (such as *Socrates*, *Indianapolis*, *Hurricane Isabel*, or *3*) and simple definite descriptions formed from the definite article *the* and a common noun (such as *the winner*, *the U.S. president*, *the park*, *the book*, or *the answer*).

In the simplest case, a “predicate” in the logical sense—for which we will use the term **predicate**—is an expression that serves to say something about the object referred to by an individual term. It is an operation whose input is the individual term and whose output is a sentence expressing what is said. So a predicate of this sort amounts to a sentence with a blank waiting to be filled by an individual term. In [6.1.2](#), we will extend this idea to include predicates that require multiple inputs (i.e., that have several blanks to be filled). Such predicates are certainly not predicates in the grammatical sense; but any predicate in the logical sense will contain the main verb of any sentence it yields as output, so many of the simplest examples of predicates will correspond to verbs or verb phrases.

So the categories of expressions we are working with now include the ones listed below (with simple examples chosen in the simple language used in some popular early elementary school

readers from the mid-20th century):

Complete expressions

sentence
<i>Jane ran and Spot barked,</i> <i>Jane ran, Spot barked</i>
individual term
<i>Jane, Spot</i>

Operations

<i>operation</i>	<i>input</i>	<i>output</i>
connective <i>__ and __</i>	sentence(s)	sentence
predicate <i>__ ran,</i> <i>__ barked</i>	individual term(s)	sentence

Since we now have a number of kinds of expression that might be input or output of an operation, there are many more sorts of operations that we might distinguish according to their input and output, and we will go on to consider some of them. For example, in [6.2.2](#), we will add a kind of operation which yields individual terms as output (for individual terms as input). And the input and output of operations need not be limited to complete expressions; in later chapters we will add operations that take predicates as input.

Glen Helman 25 Aug 2005

6.1.2. Logical predicates

We derived the concept of an individual term from a traditional description of the grammatical subject of a sentence by focusing on the semantic idea of naming. As we will see in 6.2.1, the idea of an individual term is much narrower than the idea of a grammatical subject: not every phrase that could serve as the subject of a sentence counts as an individual term.

We have seen that the opposite is true of our concept of a predicate: it includes grammatical predicates but many other expressions, too. The definition of a predicate in our sense is, like the definition of an individual term, a semantic one: a predicate says something about the about whatever objects are named by the individual terms to which it is applied. The simplest example of this is a grammatical predicate that says something about an object named by an individual term. But consider a sentence that has not only a subject but also a direct object—*Ann met Bill* for example. This says something about Ann, but it also says something about Bill. From a logical point of view, we could equally well divide the sentence into the subject *Ann* on the one hand and the predicate *met Bill* on other or into the subject-plus-verb *Ann met* and the direct object *Bill*. And we will be most in the spirit of the idea that predicates are used to say something about individuals if we divide the sentence into the two individual terms *Ann* and *Bill* on the one hand and the verb *met* on the other. The subject and object both are names, and the verb says something about the people they name. That is why we define a **predicate** as an operation that forms a sentence when applied to one or more terms. We will speak of the application of this operation as **predication** and speak of a sentence that results as **a predication**.

We can present predicates in this sense graphically by considering sentences containing any number of blanks. For example, the predication *Jane called Spot* might be depicted as follows:

Individual terms: *Jane* *Spot*
Predicate: _____ *called* _____

The number of different terms to which a predicate may be applied is its number of **places**, so the predicate above has 2 places while predicates, like _____ *ran* and _____ *barked*, that are predicates in the grammatical sense will have one place.

In the example above, the two-place predicate is a transitive verb and the second individual term functions as a direct object in the resulting sentence. The individual terms that serve as input to predicates also often appear as indirect objects or as the objects of prepositional phrases that modify a verb—as in the following examples:

Individual terms: *Jane* *Spot* *the ball*
Predicate: _____ *threw* _____

Individual terms: *the ball* *the window* *the fishbowl*
Predicate: _____ *went through* _____ *into* _____

Other examples of many-place predicates are provided by sentences containing comparative constructions or relative terms. Even conjoined subjects can indicate a many-place predicate when *and* is used to indicate the terms of a relation rather than to state a conjunction:

Individual terms: *Jane* *Sally*
Predicate: _____ *is older than* _____

Individual terms: *2* *5*
Predicate: _____ *<* _____

Individual terms: *Jane* *Sally*
Predicate: _____ *is a sister of* _____

Individual terms: *Jane* *Sally*
Predicate: _____ *and* _____ *are sisters*

Although you will rarely run into predicates with more than three or four places, it is not hard to make up examples of predicates with arbitrarily large numbers of places. For example, imagine the predicate you would get by analyzing a sentence that begins *Sam travelled from New York to Los Angeles via Newark, Easton, Bethlehem,* and goes on to state the full itinerary of a trans-continental bus trip.

The places of a many-place predicate come in a particular order. For example, the sentences *Jane is older than Sally* and *Sally is older than Jane* are certainly not equivalent, so it matters which of *Jane* and *Sally* is in the first place and which in the second when we identify them as the inputs of the predicate _____ *is older than* _____. Even when the result of reordering individual terms is equivalent to the original sentence, we will count the places as having a definite order and treat any reordering of the terms filling them as a different sentence. So *Dick is the same age as Jane* and *Jane is the same age as Dick* will count as different sentences even though _____ *is the same age as* _____ is symmetric in the sense that
$$\sigma \text{ is the same age as } \tau \Leftrightarrow \tau \text{ is the same age as } \sigma$$

for any terms σ and τ .

The only restriction on an analysis of a sentence into a predicate and individual terms is that the contribution of an individual term to the truth value of a sentence must lie only in its reference value—that is, only

in what it names if it names something and only if in the fact that it names nothing if it does not and has the nil reference value mentioned in [1.3.4](#). This means that the predicates we will consider are like truth-functional connectives in being **extensional operations**: the extension of their output depends only on the extensions of their inputs.

In the specific case of predicates, this requirement is sometimes spoken of as a requirement of **referential transparency**. When evaluating the truth-value of a sentence we sometimes look through individual terms and pay attention only to their reference values while in other cases we pay attention to the terms themselves or the ways in which they refer to their values because differences of this sort make a difference for the truth value of the sentence. For example, in deciding the truth of *The U. S. president is over 40* all that matters about the individual term *the U. S. president* is who it refers to. On the other hand, the sentence *For the past two centuries, the U. S. president has been over 35* is true while the sentence *For the past two centuries, George Bush has been over 35* is false—even when the terms *the U. S. president* and *George Bush* refer to the same person. So, in this second case, we must pay attention to differences between terms that have the same reference value. When this is so the occurrences of these terms are said to be **referentially opaque**; that is, we cannot look through them to their reference values. The restriction on the analysis of sentences into predicates and individual terms is then that we can count an occurrence of an individual term as filling the place of a predicate only when that occurrence is referentially transparent. Occurrences that are referential opaque cannot be separated from the predicate and must remain part of it.

Hints of idea of a predicate as an incomplete expression can be found in the Middle Ages but it was first presented explicitly a little over a century ago by Gottlob Frege. Frege applied the idea of an incomplete expression not only to predicates but also to mathematical expressions for functions. Indeed, Frege spoke of predicates as signs for a kind of function, a function whose value is not a number but rather a truth value. That is, just as a function like addition takes numbers as input and issues a number as output, a predicate is a sign for a function that takes the possible references of individual terms as input and issues a truth value as output. It does this by saying something true or false about its input.

We will speak of the truth-valued function associated with a 1-place predicate as a **property** and speak of the function associated with a predicate of two or more places as a **relation**. Occasionally, we will want to speak of properties and relations collectively; we will use the term **attribute** for this. Thus a predicate is a sign for an attribute in the way a truth-functional connective is a sign for a truth function.

Just as a truth-functional connective can be given a truth table, the extensionality of predicates means that a table can capture the way the truth values of their output sentences depend on the reference values of their input. For example, consider the predicate *___ divides ___ (evenly)*. Just as there can be addition or multiplication tables displaying the output of arithmetic functions for a limited range of input, we can give a table indicating the output of the relation expressed by this predicate. For the first half dozen positive integers, we would have the table shown below. Here the input for the first place of the predicate is shown by the row labels at the left and the input for the second place by the column labels at the top. The first row of the table then shows that 1 divides all six integers evenly, the second row shows that 2 divides only 2, 4, and 6 evenly, and the final column shows that each of 1, 2, 3, and 6 divides 6 evenly.

<i>divides</i>	1	2	3	4	5	6
1	T	T	T	T	T	T
2	F	T	F	T	F	T
3	F	F	T	F	F	T
4	F	F	F	T	F	F
5	F	F	F	F	T	F
6	F	F	F	F	F	T

Of course, this table does not give a complete account of the meaning of the predicate; and, for many predicates, no finite table could. But such tables like this will still be of interest to us because we will consider cases where there are a limited number of reference values and, in such cases, tables can give full accounts of predicates.

As was noted in [1.3.4](#), we assume that sentences have truth values even when they contain terms that do not refer to anything. This means that we must assume that predicates yield a truth value as output even the nil value is part of their input; that is, we assume that predicates are **total**. The truth value that is issued as output when the input includes the nil value is usually not settled by the ordinary meaning of an English predicate. It is analogous to the supplements to contexts of use suggested in [1.3.2](#) as a way of handling cases of vagueness. As in that case, we try to avoid making anything depend on the particular output in cases of undefined input but instead look at relations among sentences that hold no matter how such output is stipulated.

6.1.3. Identity

Although all the connectives that figured in our analyses of logical form received special notation and had logical properties we studied, only one predicate will count as **logical vocabulary** in this sense. Other predicates and all unanalyzed individual terms will be, like unanalyzed component sentences, part of the **non-logical vocabulary** which is assigned a meaning only by an interpretation.

The predicate that is part of our logical vocabulary will be referred to as **identity**. It is illustrated in the following sentences:

George Bush is the U.S. president

The winner was Funny Cide

$n = 3$

The morning star and the evening star are the same thing.

We will refer to such sentences as **equations**; they constitute a particular kind of predication.

In our symbolic notation, we will follow the third example and use the sign = to mark identity. As English notation, we will use the word **is**. We will represent unanalyzed individual terms by lower case letters, so we can analyze the sentences above as follows:

George Bush is the U.S. president

George Bush = the U.S. president

$g = p$

$g \text{ is } p$

[g: *George Bush*; p: *the U.S. president*]

The winner was Funny Cide

the winner = Funny Cide

$w = f$

$w \text{ is } f$

[f: *Funny Cide*; w: *the winner*]

$n = 3$

$n = t$

$n \text{ is } t$

[n: *n*; t: 3]

The morning star and the evening star are the same thing
the morning star = the evening star

$m = e$

$m \text{ is } e$

[m: *the morning star*; e: *the evening star*]

Glen Helman 25 Aug 2005

6.1.4. Abstracts

When we recognize an equation, we know immediately that it is a two-place predication; but, in other cases, identifying the places of a predicate is one of the chief tasks in analyzing a predication. A full analysis of a predication will identify one place in the predicate for each individual term appearing in the predication (more precisely, for each occurrence of an individual term satisfying the requirement referential transparency). When giving an analysis of an English sentence, it is natural also to regard the order of the places of the predicate we identify as being the same as the order in which the terms filling them appear in the English sentences. But this way of associating places of a predicate with individual terms in its English output is not required by the concept of a predicate. And, even though we will not make use of the freedom to depart from it in our analyses, we will want to allow intentional interpretations of symbolic predications to have full freedom in this regard.

For this reason, we need notation for predicates that will allow us to specify an order for the places of a predicate that is different from the order of blanks they correspond to and that will allow us to associate a given place with more than one blank. What we will use is an extension of the ordinary algebraic use of variables. It is a simple idea that was used by Frege but it was first studied extensively by the American logician Alonzo Church (1903-1995) in the 1930s, and it is his notation for it that has become standard. The usual form of definition for a function—for example,

$$f(x, y) = x^2 + 3xy + 1$$

gives a name to the function and uses a variable or variables to indicate the input values, with the output specified by an algebraic formula. Such definition might be read *f is the function which, when given input x and y, yields the output $x^2 + 3xy + 1$* . Church's notation, the notation of **lambda abstraction**, provides a symbolic version of the definite description following *is* in the English definition above. Using this notation, the symbolic definition could be written as

$$f = \lambda xy (x^2 + 3xy + 1)$$

That is, $\lambda xy (x^2 + 3xy + 1)$ can be read as *the function which,*

when given input x and y, yields the output $x^2 + 3xy + 1$. The notation of lambda abstraction thus describes a function without introducing a name for it. This idea has been important in the development of computer programming languages and, in that context, the right-hand side of the second equation taken by itself would now often be described as an “anonymous function”; so, when it is expressed in the notation of lambda abstraction, the defining equation specifies a function anonymously and then assigns it the name “f.”

Since the variables *x* and *y* might appear in any order and any number of times in the expression specifying the output, this sort of notation provides the kind of flexibility we want in specifying predicates. For example, we can write

$$\lambda xy (y \text{ told } x \text{ about } y)$$

for a predicate that, when given the input *Ann* and *Bill* yields the output *Bill told Ann about Bill*—or, more idiomatically, *Bill told Ann about himself*. As another example, note that the fullest and most natural analysis of this output sentence would instead see it as the output of the predicate

$$\lambda xyz (x \text{ told } y \text{ about } z)$$

when it is given as its input the names *Bill*, *Ann*, and *Bill* again (where the second “Bill” is the same name again and not merely the same letters; that is, it is not the name of another person).

We will refer to such an expression as an **abstract** and, more specifically as a **predicate abstract** when its output is a sentence. The general form of an abstract with *n* places is

$$\lambda x_1 \dots x_n (\dots x_1 \dots x_n \dots)$$

λ-operator *body*

It has of two parts, a **lambda operator** consisting of the letter λ followed by a list of *n* variables and, as its **body**, an expression formed drawing on these variables and other vocabulary. The variables need not actually appear in the body (to allow for lambda abstracts that achieve the effect of definitions like $f(x) = 2$); when they do appear, their occurrences in the body are said to be **bound** to the lambda operator. When the abstract is a predicate abstract, it might be read as

the attribute that $x_1 \dots x_n$ have when it is true that $\dots x_1 \dots x_n \dots$

We might take an abbreviated version of this reading as English notation for predicate abstracts:

the attribute of $x_1 \dots x_n$ that ... $x_1 \dots x_n \dots$

In case of the first predicate abstract above, we would have the following symbolic form, English reading, and English notation:

λxy (y told x about y)

the attribute that x and y have when it is true that y told x about y

the attribute of x and y that y told x about y

The English notation for abstracts is further from standard English than was the English notation we used for connectives and we will use it less often.

Variables have the grammatical status of individual terms but have no reference values. Until it is bound to a lambda operator, a variable is little more than a different way of marking a blank in a sentence. It does more than a simple line only because it indicates that the blank is one that could be linked to the place of a predicate. When it is bound, a variable functions more actively, but its function is merely to mark a correspondence between the place of a predicate and a blank in a sentence. Because of this, an older terminology referred to bound variables as “apparent variables.” And a less convenient but clearer notation would replace these variables by a more direct indication of the correspondence that they mark—e.g., by lines linking places after the initial λ with locations in the body, as in the following alternative to the first example above:

λxy (y told x about y)

λ

(told		about)
---	--	------	--	-------	--	---

In the latter diagram, lines show where occurrences of the terms serving as input should be placed in the body in order to form the output. In the lambda abstract, the same thing is indicated by the correspondence between the variables in the lambda operator and the variables marking blanks in the body.

Because bound variables only mark a correspondence between locations in the λ -operator and the body of the abstract, the bound variables of different abstracts have no connection with one another. This means that, for example, the following abstracts express the same predicate:

λxy (y told x about y)

λyz (z told y about z)

Each says that for any input terms τ and u (in that order), the output sentence should be u told τ about u .

We will refer to as **alphabetic variants** expressions, like the two abstracts above, that differ only in the particular variables they use to link a lambda operator to places in the body of an abstract. Notice that, although the variable y appears in both, it would be replaced by a different one of the input terms in each case. Because the identity and difference of variables matters only for establishing links within an abstract, there is no connection between occurrences of a variable in different abstracts.

English has devices which function like bound variables. In the general case, an abstract might be stated fully in English as follows:

$\lambda x_1 \dots x_n$ (... $x_1 \dots x_n \dots$)

the attribute that n things have when it is true that ... the first ...
the n th ...

In this form of words, the “ n things” are understood to be given as a list of (not necessarily distinct) things of length n and expressions like *the first*, *the second*, and so on, refer back to locations in this list. The description of the attribute tells when it is possessed by any such list of things, so no definite list of things is in question; and the expressions *the first*, etc., that refer back to list locations make no definite reference outside the sentence. In short, expressions like *the first* function here like pronouns. This may be clearer if we consider the case of a one-place predicate abstract along with a comparable English expression:

λx (*Tom bought* x)

the property that a thing has when it is true that Tom bought it

The English fills the blank marked by x in the body of the abstract with a pronoun *it* that has *a thing* as its antecedent. Since *a thing* makes no definite reference, neither does the pronoun; the pronoun “refers back” to its antecedent only in the sense that their references are linked in their indefiniteness: they are not indefinite in independent ways. The general moral is that the variables used in abstracts are like pronouns, and the lambda operators are like their antecedents. You should not expect variables in the scope of different lambda operators to be linked in their reference any more

than you would expect this of pronouns with different antecedents.

It is sometimes useful to consider the body of an abstract by itself. Just as a variable is grammatically like an individual term but does not have a reference value (not even the nil value), an expression like

x *told* y *about* z

that contains unbound variables is grammatically like a sentence but does not say anything. It is merely a sentence with blanks that might correspond to places of a predicate. The term **formula** is used for any expression that is grammatically like a sentence, with the term “**sentence**” reserved for formulas all of whose variables are bound (to abstracts within the sentence). Since all formulas are grammatically like sentences, the grammatical vocabulary applied to “sentences” in previous chapters applies to all formulas. In particular, formulas can be built from formulas by use of connectives, so formulas can be compound and have components.

In particular, we can speak of an **atomic formula**. Now that we analyze sentences and other formulas into components like predicates and individual terms, the atomic formulas will no longer be simply the unanalyzed sentences though those will still count as atomic. We will also count as atomic any predication. Although predications are compound and can even have formulas as components (though not as immediate components), their role in derivations is sufficiently analogous to that of unanalyzed sentences for it to make sense to put them both in the same category. There is a way of building the analogy into our syntactic categories: an unanalyzed sentence can be thought of as a **zero-place predicate**, one that requires no input to yield a sentence as output.

Glen Helman 25 Aug 2005

6.1.5. Analyzing predications

In our symbolic notation for predications of non-logical predicates—that is, for the predications that are not equations—the predicate will come first followed by the individual terms that are its input. When the predicate is expressed by an abstract, it will be enclosed in square brackets, so we might begin an analysis of *Bill told Ann about himself* as follows:

	<i>Bill told Ann about himself</i>
Identify (referentially transparent) occurrences of individual terms within the sentence, making sure they are all independent by replacing pronouns by their antecedents	<u><i>Bill</i></u> <i>told</i> <u><i>Ann</i></u> <i>about</i> <u><i>Bill</i></u>
Separate the terms from the rest of the sentence	<u><i>Bill</i></u> <u><i>Ann</i></u> <u><i>Bill</i></u> <i>told</i> <i>about</i>
Preserve the order of the terms, and establish the order of the places of the predicate by putting distinct variables in the blanks left by the terms and applying a lambda operator for these variables	λxyz (x <i>told</i> y <i>about</i> z) <i>Bill Ann Bill</i>
Surround the predicate abstract with brackets to predicate it, forming a sentence with blanks at the end	[λxyz (x <i>told</i> y <i>about</i> z)] <u> </u> <i>Bill Ann Bill</i>
Write the terms in the places of the predicate abstract	[λxyz (x <i>told</i> y <i>about</i> z)] <u><i>Bill Ann Bill</i></u>

Underlining will often be used, as it is here, to mark the places of predicates when they are filled by English expressions.

In examples and answers to exercises, we will move directly from the second of these steps to the last, so the process can be thought of as one of removing terms, placing them (in order and with any repetitions) after the sentence they are removed from, and filling the blanks left in that sentence with distinct variables, applying a corresponding lambda operator and surrounding it with brackets.

In general, an application of an n -place predicate θ to a series of n individual terms τ_1, \dots, τ_n takes the form

$$\theta\tau_1\dots\tau_n$$

and our English notation is this:

$$\theta \text{ fits } \tau_1, \dots, 'n \tau_n$$

The use of the verb *fit* here is somewhat artificial. It provides a short verb that enables $\theta\tau_1\dots\tau_n$ to be read as a sentence, and it is not too hard to understand it as saying that θ is true of τ_1, \dots, τ_n . Another artificial aspect of this notation is the unemphasized form 'n, which is designed to distinguish the use of *and* here to join the terms of a relation from its use as a truth-functional connective. We will use the general notation $\theta\tau_1\dots\tau_n$ when we wish to speak of all predications, so we will take it to apply to equations, too, even though the predicate = is written between the two terms to which it is applied.

In our fully symbolic analyses, unanalyzed non-logical predicates will be abbreviated by capital letters. This is consistent with our use of capital letters for unanalyzed sentences and with the idea that such a sentence amounts to a zero-place predicate. (When we add non-logical operations that yield individual terms as output, they will be abbreviated by lower case letters just as unanalyzed individual terms are.)

As was down in the display above, we will use the Greek letters $\theta, \pi,$ and ρ to refer to stand for any predicates, so they may stand for single letters, abstracts, or =. For the time being, all terms will be single letters in our symbolic notation; but in the next section we will consider compound terms, so we will use the Greek letters $\tau, \sigma,$ and υ to stand for any terms, simple or compound.

If we continue the analysis of *Bill told Ann about himself* into fully symbolic form, we would get the following:

Bill told Ann about himself
Bill told Ann about Bill
[$\lambda xyz (x \text{ told } y \text{ about } z)$] Bill Ann Bill
Tbab
T fits b, a, 'n b

[T: $\lambda xyz (x \text{ told } y \text{ about } z)$; a: *Ann*; b: *Bill*]

The abstract does not appear in the final analysis but it does appear in the key. The entry

$$T: \lambda xyz (x \text{ told } y \text{ about } z)$$

in the key identifies T as a predicate that, when applied to terms σ, τ, υ (in that order) yields as output the sentence $\sigma \text{ told } \tau \text{ about } \upsilon$.

When sentences contain truth-functional structure, that structure should be analyzed first; an analysis into predicates and individual terms should begin only when no further analysis by connectives is possible. Here is an example:

If either Ann or Bill was at the meeting, then Carol has seen the report and will call you about it
Either Ann or Bill was at the meeting \rightarrow *Carol has seen the report and will call you about it*
(Ann was at the meeting \vee Bill was at the meeting)
 \rightarrow (Carol has seen the report \wedge Carol will call you about the report)
([$\lambda xy (x \text{ was at } y)$] Ann the meeting \vee [$\lambda xy (x \text{ was at } y)$] Bill the meeting)
 \rightarrow ([$\lambda xy (x \text{ has seen } y)$] Carol the report \wedge [$\lambda xyz (x \text{ will call } y \text{ about } z)$] Carol you the report)

$$(Aam \vee Abm) \rightarrow (Scr \wedge Lcor)$$

if either A fits a 'n m or A fits b 'n m then both S fits c 'n r and L fits c, o, 'n r

[A: $\lambda xy (x \text{ was at } y)$; L: $\lambda xyz (x \text{ will call } y \text{ about } z)$; S: $\lambda xy (x \text{ has seen } y)$; a: *Ann*; b: *Bill*; c: *Carol*; m: *the meeting*; o: *you*; r: *the report*]

When analyzing atomic sentences into predicates and terms be sure to watch for repetitions of predicates from one atomic sentence to another; such repetitions are an important part of the logical structure of the sentence.

Since the notation for identity is different from that used for non-logical predicates, you need to watch for atomic sentences that count as equations. These will usually, but not always, be marked by some form of the verb *to be* but, of course, forms of *to be* have other uses, too. Consider the following example:

If Tom was told of the nomination, then if he was the winner he wasn't surprised
Tom was told of the nomination \rightarrow *if Tom was the winner he wasn't surprised*

Tom was told of the nomination \rightarrow (*Tom was the winner*
 \rightarrow *Tom wasn't surprised*)

Tom was told of the nomination \rightarrow (*Tom was the winner*
 \rightarrow \neg *Tom was surprised*)

$[\lambda xy (x \text{ was told of } y)]$ *Tom the nomination*
 \rightarrow (*Tom = the winner* \rightarrow \neg $[\lambda x (x \text{ was surprised})]$ *Tom*)

$Ltn \rightarrow (t = r \rightarrow \neg St)$

if L fits t 'n n then if t is r then not S fits t

[L: $\lambda xy (x \text{ was told of } y)$; S: $\lambda x (x \text{ was surprised})$; t: *Tom*;
n: *the nomination*]

It is fairly safe to assume that a form of *to be* joining to individual terms indicates an equation, but it is wise to always think about what is being said: an equation is a sentence that says its component individual terms have the same reference value. Notice also that identity does not appear in the key to the analysis. That is because it is part of the logical vocabulary; that is, it is like the connectives, which also do not appear in keys.

Glen Helman 25 Aug 2005

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 $\lambda x_1 \dots x_n (\dots x_1 \dots x_n \dots)$ —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** $x_1 \dots x_n$ **that** $\dots x_1 \dots x_n \dots$. 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 $\tau_1, \dots, 'n \tau_n$.

Glen Helman 25 Aug 2005

6.1.x. Exercise questions

1. Analyze each of the following sentences in as much detail as possible.
 - a. *Ann introduced Bill to Carol.*
 - b. *Ann gave the book to either Bill or Carol.*
 - c. *Ann gave the book to Bill and he gave it to Carol.*
 - d. *Tom had the package sent to Sue, but it was returned to him.*
 - e. *Georgia will see Ed if she gets to Denver before Saturday.*
 - f. *If the murderer is either the butler or the nephew, then I'm Sherlock Holmes.*
 - g. *Neither Ann nor Bill saw Tom speak to either Mike or Nancy.*
 - h. *Tom will agree if each of Ann, Bill, and Carol asks him.*

2. Synthesize idiomatic English sentences that express the propositions associated with the logical forms below by the intensional interpretations that follow them.
 - a. $Wci \wedge Sc1$
[S: λxy (x *is south of* y); W: λxy (x *is west of* y); c: *Crawfordsville*; i: *Indianapolis*; l: *Lafayette*]
 - b. $Mab \rightarrow Mba$
[M: λxy (x *has met* y); a: *Ann*; b: *Bill*]
 - c. $Iacb \wedge Iadb$
[I: λxyz (x *introduced y to* z); a: *Alice*; b: *Boris*; c: *Clarice*; d: *Doris*]
 - d. $Wab \wedge Kabab$
[K: $\lambda xyzw$ (x *asked y to write z about* w); W: λxy (x *wrote to* y); a: *Alice*; b: *Boris*]
 - e. $g = c \rightarrow (f = s \wedge p = t)$
[c: *the city*; f: *football*; g: *Green Bay*; p: *the Packers*; s: *the sport*; t: *the team*]

Glen Helman 25 Aug 2005

6.1.xa. Exercise answers

1. a. Ann introduced Bill to Carol
 $[\lambda xyz (x \text{ introduced } y \text{ to } z)] \underline{\text{Ann Bill Carol}}$
 Iabc
 I fits a, b, 'n c
 [I: $\lambda xyz (x \text{ introduced } y \text{ to } z)$; a: Ann; b: Bill; c: Carol]
- b. Ann gave the book to either Bill or Carol
Ann gave the book to Bill \vee Ann gave the book to Carol
 $[\lambda xyz (x \text{ gave } y \text{ to } z)] \underline{\text{Ann the book Bill}} \vee [\lambda xyz (x \text{ gave } y \text{ to } z)] \underline{\text{Ann the book Carol}}$
 Gakb \vee Gake
 either G fits a, k, 'n b or G fits a, k, 'n c
 [G: $\lambda xyz (x \text{ gave } y \text{ to } z)$; a: Ann; b: Bill; c: Carol; k: the book]
- c. Ann gave the book to Bill and he gave it to Carol
Ann gave the book to Bill \wedge Bill gave the book to Carol
 $[\lambda xyz (x \text{ gave } y \text{ to } z)] \underline{\text{Ann the book Bill}} \wedge [\lambda xyz (x \text{ gave } y \text{ to } z)] \underline{\text{Bill the book Carol}}$
 Gakb \wedge Gbkc
 both G fits a, k, 'n b and G fits b, k, 'n c
 [G: $\lambda xyz (x \text{ gave } y \text{ to } z)$; a: Ann; b: Bill; c: Carol; k: the book]
- d. Tom had the package sent to Sue, but it was returned to him
Tom had the package sent to Sue \wedge the package was returned to Tom
 $[\lambda xyz (x \text{ had } y \text{ sent to } z)] \underline{\text{Tom the package Sue}} \wedge [\lambda xy (x \text{ was returned to } y)] \underline{\text{the package Tom}}$
 Htps \wedge Rpt
 both H fits t, p, 'n s and R fits p 'n t
 [H: $\lambda xyz (x \text{ had } y \text{ sent to } z)$; R: $\lambda xy (x \text{ was returned to } y)$; p: the package; s: Sue; t: Tom]
- e. Georgia will see Ed if she gets to Denver before Saturday
Georgia will see Ed \leftarrow Georgia will get to Denver before Saturday
 $[\lambda xy (x \text{ will see } y)] \underline{\text{Georgia Ed}} \leftarrow [\lambda xyz (x \text{ will get to } y \text{ before } z)] \underline{\text{Georgia Denver Saturday}}$

- Sge \leftarrow Ggds
 Ggds \rightarrow Sge
 if G fits g, d, 'n s then S fits g 'n e
 [G: $\lambda xyz (x \text{ will get to } y \text{ before } z)$; S: $\lambda xy (x \text{ will see } y)$; d: Denver; e: Ed; g: Georgia; s: Saturday]
- f. If the murderer is either the butler or the nephew, then I'm Sherlock Holmes
the murderer is either the butler or the nephew \rightarrow I'm Sherlock Holmes
 $(\underline{\text{the murderer is the butler}} \vee \underline{\text{the murderer is the nephew}}) \rightarrow \underline{I = \text{Sherlock Holmes}}$
 $(\underline{\text{the murderer}} = \underline{\text{the butler}} \vee \underline{\text{the murderer}} = \underline{\text{the nephew}}) \rightarrow i = s$
 $(m = b \vee m = n) \rightarrow i = s$
 if either m is b or m is n then i is s
 [b: the butler; i: I; m: the murderer; n: the nephew; s: Sherlock Holmes]
- g. Neither Ann nor Bill saw Tom speak to either Mike or Nancy
 $\neg (\underline{\text{Ann saw Tom speak to either Mike or Nancy}} \vee \underline{\text{Bill saw Tom speak to either Mike or Nancy}})$
 $\neg ((\underline{\text{Ann saw Tom speak to Mike}} \vee \underline{\text{Ann saw Tom speak to Nancy}}) \vee (\underline{\text{Bill saw Tom speak to Mike}} \vee \underline{\text{Bill saw Tom speak to Nancy}}))$
 $\neg (([\lambda xyz (x \text{ saw } y \text{ speak to } z)] \underline{\text{Ann Tom Mike}} \vee [\lambda xyz (x \text{ saw } y \text{ speak to } z)] \underline{\text{Ann Tom Nancy}}) \vee ([\lambda xyz (x \text{ saw } y \text{ speak to } z)] \underline{\text{Bill Tom Mike}} \vee [\lambda xyz (x \text{ saw } y \text{ speak to } z)] \underline{\text{Bill Tom Nancy}}))$
 $\neg ((\text{Satm} \vee \text{Satn}) \vee (\text{Sbtm} \vee \text{Sbtn}))$
 not either either S fits a, t, 'n m or S fits a,t, 'n n or either S fits b,t, 'n m or S fits b,t, 'n n
 [S: $\lambda xyz (x \text{ saw } y \text{ speak to } z)$; a: Ann; b: Bill; m: Mike; n: Nancy; t: Tom]
- h. Tom will agree if each of Ann, Bill, and Carol asks him
Tom will agree \leftarrow each of Ann, Bill, and Carol will ask Tom
Tom will agree $\leftarrow ((\underline{\text{Ann will ask Tom}} \wedge \underline{\text{Bill will ask Tom}}) \wedge \underline{\text{Carol will ask Tom}})$
 $[\lambda x (x \text{ will agree})] \underline{\text{Tom}} \leftarrow (([\lambda xy (x \text{ will ask } y)] \underline{\text{Ann Tom}}$

$\wedge [\lambda xy (x \text{ will ask } y)] \underline{\text{Bill Tom}} \wedge [\lambda xy (x \text{ will ask } y)]$
Carol Tom)

$Gt \leftarrow ((Aat \wedge Abt) \wedge Act)$
 $((Aat \wedge Abt) \wedge Act) \rightarrow Gt$

if both both A fits a 'nt and A fits b 'nt and A
fits c 'nt then G fits t

[A: $\lambda xy (x \text{ will ask } y)$; G: $\lambda x (x \text{ will agree})$; a: *Ann*; b: *Bill*;
c: *Carol*; t: *Tom*]

The function of *each* here is to indicate a group of two-
place predication rather than a single four-place predicate
 $\lambda xyzw (x, y, \text{ and } z \text{ will ask } w)$, which is what would be
required in order to express instead the idea of Ann, Bill,
and Carol making the request as a group.

2. a. $[\lambda xy (x \text{ is west of } y)] \underline{\text{Crawfordsville Indianapolis}}$
 $\wedge [\lambda xy (x \text{ is south of } y)] \underline{\text{Crawfordsville Lafayette}}$
Crawfordsville is west of Indianapolis \wedge *Crawfordsville*
is south of Lafayette
Crawfordsville is west of Indianapolis and south of
Lafayette
- b. $[\lambda xy (x \text{ has met } y)] \underline{\text{Ann Bill}} \rightarrow [\lambda xy (x \text{ has met } y)] \underline{\text{Bill}}$
Ann
Ann has met Bill \rightarrow *Bill has met Ann*
If Ann has met Bill then he has met her
- c. $[\lambda xyz (x \text{ introduced } y \text{ to } z)] \underline{\text{Alice Clarice Boris}}$
 $\wedge [\lambda xyz (x \text{ introduced } y \text{ to } z)] \underline{\text{Alice Doris Boris}}$
Alice introduced Clarice to Boris \wedge *Alice introduced Doris*
to Boris
Alice introduced Clarice and Doris to Boris
- d. $[\lambda xy (x \text{ wrote to } y)] \underline{\text{Alice Boris}}$
 $\wedge [\lambda xyzw (x \text{ asked } y \text{ to write } z \text{ about } w)] \underline{\text{Alice Boris}}$
Alice Boris
Alice wrote to Boris \wedge *Alice asked Boris to write Alice*
about Boris
Alice wrote to Boris \wedge *Alice asked Boris to write her*
about himself
Alice wrote to Boris and asked him to write her about
himself
- e. $g = c \rightarrow (f = s \wedge p = t)$
Green Bay = the city \rightarrow (football = the sport \wedge the Packers

= the team)

Green Bay is the city \rightarrow (*football is the sport* \wedge *the*
Packers are the team)

Green Bay is the city \rightarrow *football is the sport and the*
Packers are the team

If Green Bay is the city, then football is the sport and the
Packers are the team

Glen Helman 25 Aug 2005