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 variety of grammatical categories can be defined using the idea of an individual term, an expression whose function is to name an individual object.

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. Extensionality
The truth value of a sentence in which a predicate is applied depends only on the reference values of the terms the predicate is applied to, so the meaning of predicate is a function from reference values to truth values.

6.1.4. Identity
We will study the special logical properties of only one predicate, the one expressed by the equals sign and by certain uses of the English word `is`.

6.1.5. Analyzing predications
When the analysis of truth-functional structure is complete, we may go on to analyze atomic sentences as the applications of predicates to individual terms.

6.1.6. Individual terms
While individual terms are not limited to proper names, they do not include all noun phrases, only ones whose function is like that of proper names.

6.1.7. Functors
Individual terms can be formed from other individual terms by operations analogous to predicates.

6.1.8. Examples and problems
These operations enable us to continue the analysis of sentences beyond the analysis of predications by analyzing individual terms themselves.

Glen Helman 11 Oct 2010

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 turn to 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.

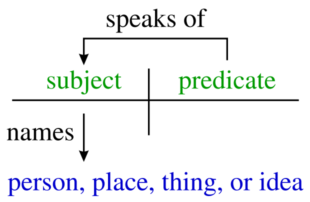


Fig. 6.1.1-1. The traditional picture of grammatical subjects and predicates.

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 still 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 and the output consists of 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; it is an expression which can be described, roughly, as naming a "person, place, thing, or idea." In 6.1.6, we will consider the full range of expressions that count as individual terms but, for now, it will be enough to have in mind two basic kinds of example—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—and this is what we will use the term *predicate* to speak of—is an expression that can be used to say something about the object referred to by an individual term. It is an operation whose input is an individual term and whose output is a sentence expressing what is said. Thus a logical predicate amounts to a sentence with a blank waiting to be filled by an individual term. In 6.1.2, we will move beyond this simple case 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; nonetheless a logical predicate 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.

The categories of expressions we are working with now include the ones listed below (with simple examples in the style of some popular early elementary school readers from the mid-20th century):

Complete expressions

expression	*examples*
sentence	Jane ran, Spot barked, Jane ran and Spot barked
individual term	Jane, Spot

Operations

operation	*input*	*output*	*examples*
connective	sentence(s)	sentence	_ and _
predicate	individual term(s)	sentence	_ ran, _ barked

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 can be distinguished according to their input and output, and we will go on to consider some of them. For example, in 6.1.7, we will add a kind of operation which yields individual terms as output (for individual terms as input). The input and output of operations need not be limited to complete expressions, and in later chapters, we will add operations that take predicates as input.

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 6.1.6, 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.

Like the definition of an individual term, the definition of a logical predicate is semantic: 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 [_ called _] has 2 places while predicates, like [_ ran] and [_ barked], that are also predicates in the grammatical sense will have one place. We will discuss our notation for predicates more in 6.2.1, but we will often (as has been done here) indicate a predicate by surrounding with brackets the English sentence-with-blanks that expresses it.

In the example above, the two-place predicate is a transitive verb and the second individual term functions as its direct object in the resulting sentence. The individual terms that serve as input to predicates may also 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 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

$$\sigma \text{ is the same age as } \tau \simeq \tau \text{ is the same age as } \sigma$$

for any terms σ and τ.

6.1.3. Extensionality

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, all that matters is what a term names if it names something; and, if it names nothing and thus has the nil reference value mentioned in 1.3.7, that is all that matters. Both truth values and reference values are extensions in the sense introduced in 2.18, so 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 it is satisfied, we can look through individual terms and pay attention only to their reference values when judging whether a sentence is true or false; in other cases, we might need to pay attention to the terms themselves or to the ways in which they refer to their values in order to judge the truth value. 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, Barack Obama has been over 35 is false—even when the terms the U. S. president and Barack Obama 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 separate an individual term from a predicate and count it as filling a place of the 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 developed explicitly by Gottlob Frege in the late 19th century. 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 + 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 by saying something true or false about the 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*. Thus a predicate is a sign for a property or relation 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 the 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 some of 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.

_ divides _	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 1.3.7, 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 1.3.6 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.

6.1.4. Identity

We used special notation for all the connectives that figured in our analyses of logical form, and they all had logical properties that we studied. However, 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*, and they will be assigned meanings only when we specify an interpretation of this vocabulary.

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

$\underline{\text{Barack Obama}}$ is $\underline{\text{the U.S. president}}$

$\underline{\text{The winner}}$ was $\underline{\text{Funny Cide}}$

$\underline{\text{n}}$ = $\underline{\text{3}}$

$\underline{\text{The morning star}}$ and $\underline{\text{the evening star}}$ are the same thing.

Sentences like these are *equations*. Equations are thus a special 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:

Barack Obama is the U.S. president
Barack Obama = the U.S. president

o = p
o is p

o: Barack Obama; p: the U.S. president

The winner was Funny Cide
the winner = Funny Cide

w = f
w is f

f: Funny Cide; w: the winner

---

n = 3

n = t
n 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 is e

m: the morning star; e: the evening star

Once in symbolic form, these equations are very simple. The greater complexity found in most interesting mathematical equations is due to the complexity of the individual terms they contain. To exhibit that complexity in our analyses, we will need to analyze individual terms, something we will begin to do in 6.1.7.

Glen Helman 03 Aug 2010

6.1.5. Analyzing predications

Apart from the special case of equations, our symbolic notation for predications will identify the predicate first followed by a list of the individual terms that are its input. This is a departure from English word order in most cases, but we can present analyses in this way even before we introduce symbols. The example below presents the analysis of a predication into a predicate and individual terms as a series of steps.

	Bill introduced himself to Ann
Identify (referentially transparent) occurrences of individual terms within the sentence, making sure they are all independent by replacing pronouns by their antecedents	Bill introduced Bill to Ann
Separate the terms from the rest of the sentence	Bill Bill Ann ___ introduced ___ to ___
Preserve the order of the terms, and form a predicate from the remainder of the sentence	Bill Bill Ann [_ introduced _ to _] ___ ___ ___
Write the terms in the places of the predicate	[_ introduced _ to _] Bill Bill Ann

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 enclosing sentence-with-blanks in brackets.

In general, an application of an *n*-place predicate θ to a series of *n* individual terms $\tau_1, \ldots, \tau_n$ takes the form

$$\theta\tau_1\ldots\tau_n$$

and our English notation is this:

θ fits (series) $\tau_1, \ldots,$ ən τ_n

The use of the verb fit here is somewhat artificial. It provides a short verb that enables $\theta\tau_1\ldots\tau_n$ to be read as a sentence, and it is not too hard to understand it as saying that θ is true of $\tau_1, \ldots, \tau_n$. Another artificial aspect of this notation is the unemphasized form ən of and, 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. The role of the term series, which will rarely be needed, is discussed in 6.1.7. We will use the general notation $\theta\tau_1\ldots\tau_n$ 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 since predicates have sentences as their output. 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 done in the display above, we will use the Greek letters θ, π, μ, and ρ to refer to stand for any predicates, so they may stand for single letters and for =. The may also stand for complex predicates whose internal structure has been analyzed, something we will go on to consider in 6.2.1. We will also go on to consider compound terms, and we will use the Greek letters τ, σ, and υ to stand for any terms, simple or compound.

If we complete the analysis of Bill introduced himself to Ann, carrying it into fully symbolic form and restating it in English notation, we would get the following:

Bill introduced himself to Ann
Bill introduced Bill to Ann
[_ introduced _ to _] Bill Bill Ann

Tbba
T fits b, b, ən a

T: [_ introduced _ to _]; a: Ann; b: Bill

Notice that the bracketed English sentence-with-blanks does not appear in the final analysis, but it does appear in the key.

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 → Carol has seen the report and will call you about it
(Ann was at the meeting ∨ Bill was at the meeting) → (Carol has seen the report ∧ Carol will call you about the report)
([_ was at _] Ann the meeting ∨ [_ was at _] Bill the meeting) → ([_ has seen _] Carol the report ∧ [_ will call _ about _] Carol you the report)

$$(Aam \lor Abm) \to (Scr \land Lcor)$$

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

A: [_ was at _]; L: [_ will call _ about _]; S: [_ has seen _]; 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 as that of [_ was at _] in this example. 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 → if Tom was the winner he wasn't surprised

Tom was told of the nomination → (Tom was the winner → Tom wasn't surprised)

Tom was told of the nomination → (Tom was the winner → ¬ Tom was surprised)

[_ was told of _] Tom the nomination
→ (Tom = the winner → ¬ [_ was surprised] Tom)

Ltn → (t = r → ¬ St)

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

L: [_ was told of _]; S: [_ was surprised]; t: Tom; n: the nomination

It is fairly safe to assume that a form of to be joining two 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. A use of to be joining noun phrases will indicate an equation only when these noun phrases are individual terms; the conditions under which that is so are discussed in the next subsection. Finally, notice that no identity predicate should appear in the key to the analysis. That is because it is part of the logical vocabulary; as such, it is like the connectives, which also do not appear in keys.

6.1.6. Individual terms

The chief examples of individual terms are proper names, for the central function of a proper name is to refer to the bearer of the name. But a proper name is not the only sort of expression that refers to an individual; a phrase like the first U. S. president serves as well as the name George Washington. In general, descriptive phrases coupled with the definite article the at least purport to refer of individuals. These phrases are the *definite descriptions* discussed briefly in 1.3.7, and we have been counting them as individual terms. Still other examples of individual terms can be found in nouns and noun phrases modified by possessives—for example, Mt. Vernon's most famous owner. Indeed, expressions of this sort can generally be paraphrased by definite descriptions (such as the most famous owner of Mt. Vernon). A final group of examples are demonstrative pronouns this and that and other pronouns whose references are determined by the context of use—such as I, you, and certain uses of third person pronouns. On the other hand, while *anaphoric pronouns*—i.e., pronouns that have other noun phrases as their antecedents—count grammatically as individual terms, they do not have independent reference values and will be treated differently in our analyses. We will look at their role more closely in 6.2.3; for now, it is enough to note that they raise issues for the analysis of predications that are analogous to the issues they raise for the analysis of truth-function compounds.

There is no traditional grammatical category or part of speech that includes individual terms but no other expressions. In particular, the class of nouns and noun phrases is too broad because it includes simple common nouns, such as president, as well as *quantifier phrases*—such as no president, every president, or a president. And neither common nouns nor quantifier phrases make the kind of reference that is required for an individual term.

Even before we look further at the reasons why this is so, we can distinguish individual terms from other nouns and noun phrases by thinking of them as answers to a which question. If you are asked Which person, place, thing, or idea are you referring to? and you reply with any of the individual terms, you have answered the question directly. On the other hand, a common noun by itself is ungrammatical as an answer, and a quantifier phrase does not provide a direct answer. While a president, no president, and every president are grammatical replies to the question Which person are you referring to?, the first two provide only an incomplete or evasive answers, and the third indicates that the question cannot be answered as asked.

The following table collects the examples we have just seen on both sides of the line between individual terms and other noun phrases:

Individual terms	*Not individual terms*
proper names	common nouns
George Washington	president
definite descriptions	quantifier phrases
the first U. S. president	no president, every president, a president
noun phrases with possessive modifiers	
Mt. Vernon's most famous owner	
non-anaphoric pronouns	
this, you	
anaphoric pronouns	
he, she, it	

Perhaps the most that can be done in general by way of defining the idea of an *individual term* is to give the following rough semantic description: an individual term is

an expression that refers (or purports to refer)
to a single object in a definite way

At any rate, this formula can be elaborated to explain the reasons for rejecting the noun phrases at the right of the table above.

The formula is intended as a somewhat more precise statement of the idea that an individual term "names a person, place, thing or idea." It uses object in place of the list person, place, thing, or idea partly for compactness and partly because that list is incomplete. Indeed it would be hard to ever list all the kinds of things that might be referred to by individual terms. If the term *object* and other terms like *entity*, *individual*, and *thing* are used in a broad abstract sense, they can apply to anything that an individual term might refer to. In particular, in this sort of usage, these terms apply to people. The main force of the formula above then lies in the ideas of *referring to a single thing* and *referring in a definite way*.

The requirement that reference be to a single thing rules out most of noun phrases on the right of the table above. First of all, if a common noun by itself can be said to refer at all, it refers not to a single thing but to a class, such as the class of all presidents. Now this class can be thought of as a single thing and can be referred to by the definite description just used—i.e., the class of all presidents—but the common noun president "refers" to this class in a different way. Common nouns are sometimes labeled *general terms* and distinguished from *singular terms*, an alternative label for individual terms. The function of a general term is to indicate a general kind (e.g., dogs) from which individual things may be picked out rather than to pick out a single thing of that kind (e.g., Spot), as an individual term does. Thus the individual term the first U. S. president picks out an individual within the class indicated by the common noun president; and the class of all presidents picks out an individual within the class indicated by the common noun class. That is, a general term indicates a range of objects from which a particular object might be chosen while an individual term picks out a particular object. Although there is much that might be said about the role of general terms in deductive reasoning, we will never identify them as separate components in our analyses of logical form, and the word term without qualification will be used as an abbreviated alternative to individual term.

The remaining noun phrases at the right of the table are like individual terms in making use of a common noun's indication of a class of objects. However, they do not do this to pick out a single member of the class but instead to help make claims about the class as a whole. The claims to which they contribute say something about the number of members of a class that have or lack a certain property, and that is the reason for describing them as "quantifier" phrases.

It's probably clear that the phrases every president and no president, even though they are grammatically singular, do not serve the function of picking out a single object. But that may be less clear in the case of a president. Sentences containing quantifier phrases like a president and some president share with those containing definite descriptions, such as the president, the feature that they can be true because of a fact about a single object. For example, The first U. S. president wore false teeth and A president wore false teeth can be said to both be true because of a fact about Washington. The difference between the two sorts of expression can be seen by considering what might make such sentences false. If Washington had not worn false teeth, The first U. S. president wore false teeth would be false but A president wore false teeth might still be true. That's because the second could be true because of facts about many different presidents (in many different countries), so its truth is not tied to facts about any one of them. If the expression a president is thought of as referring at all, its reference is an indefinite one. That is one reason for adding the qualification definite to the formula for individual terms given above, but this qualification also serves as a reminder that the presence of a definite article is a mark of an individual term while an indefinite article indicates a quantifier phrase.

Glen Helman 11 Oct 2010

6.1.7. Functors

Truth-functional connectives express truth-valued functions of truth values, and predicates express truth-valued functions of reference values. A third sort of function not only takes reference values as input but also issues them as output. We will refer to this sort of function as a *reference function* or, in contexts where we do not need a more general concept, simply as a *function*. We will refer to expressions that are signs for these functions as *functors* and refer to the operation of applying a functor as *function application*. We can speak of the result of a function application as a *compound term*.

Functors are incomplete expressions that stand to individual terms as connectives stand to sentences, so we can extend the table of operations in 6.1.1 as follows:

operation	*input*	*output*
connective	sentence(s)	sentence
predicate	individual term(s)	sentence
functor	individual term(s)	individual term

We will add further incomplete expressions to this list in later chapters when we consider operations that take predicates as input.

Signs for mathematical functions provide examples of functors. The expression 7 + 5 can be analyzed as

Individual terms: 7 5
Functor: ___ + ___

But functors are not limited to mathematical vocabulary. Any individual term that contains one or more individual terms can be seen as the result of applying a functor to those component terms. Thus the oldest child of Ann and Bill can be analyzed as

Individual terms: Ann Bill
Functor: the oldest child of ___ and ___

And the more complex individual term the book that Ann's father mentioned has the following analysis:

Individual term: Ann
Functors: ___'s father
the book that ________ mentioned

Possessives and prepositional phrases often give rise to functors but all that is needed to have a functor is an individual term that contains an individual term.

Our notation for functors will be analogous to that for predicates. Functors can be represented in semi-symbolic notation by individual-terms-with-blanks surrounded by brackets. Using this notation, the first two examples above could be given the analyses:

[_ + _] 7 5
[the oldest child of _ and _] Ann Bill

In the case of the third example, we will use parentheses to show grouping

[the book that _ mentioned] ([_'s father] Ann)

In fact, there is no danger of ambiguity here; but the structure is clearer with parentheses, and, in the full symbolic notation, compound terms should be enclosed in parentheses when they fill a place of a functor or predicate.

In that notation, unanalyzed functors will be represented by lower case letters and will be written before the individual terms filling their places. The general form of a compound term is this

$$\zeta\tau_1\ldots\tau_n$$

and our English notation will be

ζ of (series) τ_1, …, ən τ_n

or

ζ applied to (series) τ_1, …, ən τ_n

both of which are in keeping with the usual way of reading a function application, but one or the other will work better in certain contexts. When we need a general variable for functors we will use ζ or ξ.

Using this symbolic and English notation, we can express the final analyses of the examples above as follows:

symbolic notation	*English notation*	*key*
psf	p of s ən f	p: [_ + _]; f: 5; s: 7
oab	o of a ən b	o: [the oldest child of _ and _]; a: Ann; b: Bill
b(fa)	b of f of a	b: [the book that _ mentioned]; f: [_'s father]; a: Ann

The symbolic notation for functors that is used here is designed to minimize parentheses and commas and is fairly common in work on logic, but it is different from the most common mathematical notation for function applications. The general rule for interpreting it is this: (i) after a predicate —i.e., after a capital letter—each unparenthesized letter and each parenthetical unit occupies one place of the predicate and (ii) within a parenthetical unit the first letter is a functor and each following unparenthesized letter and each parenthetical unit occupies one place of this functor.

Here are some examples for comparison

common mathematical notation	*symbolic notation used here*	*English notation*
f(a)	fa	f of a
f(a, b)	fab	f of a ən b
f(g(a))	f(ga)	f of g of a
f(a, g(b))	fa(gb)	f of a ən g of b
f(g(a), b)	f(ga)b	f of series g of a ən b
f(g(a, b))	f(gab)	f of g of series a ən b

The last two examples above show the role of the optional term series in avoiding ambiguity. Because the letters used to represent functors and non-logical predicates do not have a fixed number of places associated with them, when a single ən follows two occurrences of of, it can be unclear where the series of terms marked by ən actually began. There are other ways of handling this ambiguity. Parentheses suffice in written notation and parentheses, like other punctuation, can be reflected in speech. For example, it is natural to mark the difference between f of (g of a) ən b and f of (g of a ən b), respectively, by varying the speed with which they are spoken in ways that might be indicated by "f of g-of-a ən b" and "f of g of a-ən-b".

In the presence of functors, the potential for undefined terms increases considerably. Even if the cat on the mat has a non-nil reference value, the cat on the refrigerator may not—to say nothing of the cat on the house of Ann's father's best friend or the cat on 6. That is, functors accept a large variety of inputs and can be expected to issue output with undefined reference for some of them. This problem can be reduced (though not eliminated) by limiting functors to input of certain sorts. That is usually done by assigning individual terms to various *types* and allowing only individual terms of certain types to serve as inputs to a given functor. For example, the functor [_ + _] might be restricted to numerical input. We will not follow this approach (which complicates the description of logical forms considerably), but it does capture a number of features, both syntactic and semantic, of a natural language like English.

6.1.8. Examples and problems

We will begin with a couple of extended but straightforward examples.

If Dan is the winner and Portugal is the place he would most like to visit, he will visit there before long

Dan is the winner and Portugal is the place he would most like to visit → Dan will visit Portugal before long

(Dan is the winner ∧ Portugal is the place Dan would most like to visit) → Dan will visit Portugal before long

($\underline{\text{Dan}}$ is $\underline{\text{the winner}}$ ∧ $\underline{\text{Portugal}}$ is $\underline{\text{the place Dan would most like to visit}}$) → $\underline{\text{Dan}}$ will visit $\underline{\text{Portugal}}$ before long

($\underline{\text{Dan}}$ = $\underline{\text{the winner}}$ ∧ $\underline{\text{Portugal}}$ = the place $\underline{\text{Dan}}$ would most like to visit) → [_ will visit _ before long] $\underline{\text{Dan}}$ $\underline{\text{Portugal}}$

(d = n ∧ p = [the place _ would most like to visit] $\underline{\text{Dan}}$) → Vdp

(d = n ∧ p = ld) → Vdp

if both d is n and p is l of d then V fits d ən p

V: [_ will visit _ before long]; l: [the place _ would most like to visit]; d: Dan; n: the winner; p: Portugal

Al won't sign the contract Barb's lawyer made out without speaking to his lawyer

¬ Al will sign the contract Barb's lawyer made out without speaking to his lawyer

¬ (Al will sign the contract Barb's lawyer made out ∧ ¬ Al will speak to his lawyer)

¬ ($\underline{\text{Al}}$ will sign $\underline{\text{the contract Barb's lawyer made out}}$ ∧ ¬ $\underline{\text{Al}}$ will speak to $\underline{\text{Al's lawyer}}$)

¬ ([_ will sign _] $\underline{\text{Al}}$ $\underline{\text{the contract Barb's lawyer made out}}$ ∧ ¬ [_ will speak to _] $\underline{\text{Al}}$ $\underline{\text{Al's lawyer}}$)

¬ (S a (the contract $\underline{\text{Barb's lawyer}}$ made out) ∧ ¬ P a ($\underline{\text{Al's lawyer}}$))

¬ (S a ([the contract _ made out] $\underline{\text{Barb's lawyer}}$) ∧ ¬ P a ([_'s lawyer] $\underline{\text{Al}}$))

¬ (S a (c ([_'s lawyer] $\underline{\text{Barb}}$)) ∧ ¬ Pa(la))

¬ (Sa(c(lb)) ∧ ¬ Pa(la))

not both S fits a ən c of l of b and not P fits a ən l of a

P: [_ will speak to _]; S: [_ will sign _]; c: [the contract _ made out]; l: [_'s lawyer]; a: Al; b: Barb

When analyzing either a predication or an individual term, make sure that you remove all the largest individual terms it contains. That is, if you identify a component individual term, make sure that it is not part of a compound term that is itself a component of the sentence or term you are analyzing. To analyze Al will speak to his lawyer as [_ will speak to _'s lawyer] $\underline{\text{Al}}$ $\underline{\text{Al}}$ would be to ignore an important aspect of its structure. Of course, when applying this maxim, it is important to distinguish individual terms from other noun phrases. For example, although Dan is the winner of the contest can be analyzed initially as $\underline{\text{Dan}}$ = $\underline{\text{the winner of the contest}}$, the grammatically similar sentence Dan is a winner of the contest should be analyzed as [_ is a winner of _] $\underline{\text{Dan}}$ $\underline{\text{the contest}}$ because a winner of the contest is not an individual term.

Also, when you locate a definite description, make sure that you have identified the whole of it. What you are most likely to miss are modifiers, usually prepositional phrases or relative clauses, that follow the main common noun of the definite description. For example, although the place might be an individual term in its own right in other cases, in the example above is it only part of the term the place Dan would most like to visit. Similarly, the contract is only the beginning of the individual term the contract Barb's lawyer made out. In both of the these cases, the rest of the definite description is a relative clause with a suppressed relative pronoun; that is, they might have been stated more fully as the place that Dan would most like to visit and the contract that Barb's lawyer made out, respectively. It might help here to think of prepositional phrases and relative clauses as modifying a common noun before the definite article is attached. That is, the phrases above have the form the (place Dan would most like to visit) and the (contract Barb's lawyer made out), so any component of these sentences containing the initial the must also contain the whole of the following parenthesized expressions.

There are some cases where a prepositional phrase or relative clause following a common noun should not be counted as part of a definite description. Some prepositional phrases can modify both nouns and verbs, and a prepositional phrase following a noun within a grammatical predicate might be understood to modify either it or the main verb. The sentence The dog chased the cat on the mat is ambiguous in this way since the mat might be understood to be either the location of the chase or the location of the cat, who might have been chased elsewhere. This sort of ambiguity can be clarified by converting the prepositional phrase into a relative clause, which can only modify a noun; if this transformation—e.g.,

The dog chased the cat that is on the mat

—preserves meaning, then the prepositional phrase is part of the definite description. On the other hand, since anaphoric pronouns cannot accept modifiers, replacing a possible noun phrase by a pronoun will produce a sentence in which a prepositional phrase unambiguously modifies the verb. This can be done by moving the noun phrase to the front of the sentence, joining it to the remaining sentence-with-a-blank by the phrase is such that, and filling the blank with an appropriate pronoun (he, she, or it). In this example, that would give us

The cat is such that the dog chased it on the mat

So, the prepositional phrase on the mat should be taken to modify cat or chased depending on whether the first or second of the displayed sentences best captures the meaning of the original. Of course, when a potentially ambiguous sentence is taken out of context, it may not be clear which of two alternatives does best capture the original meaning; in such a case, either analysis is a possible interpretation and the difference between them shows what further information is needed in order to determine what was meant.

Not all relative clauses contribute to determining reference. Those that do are *restrictive* clauses, and it is these that should be included in definite descriptions. Other relative clauses are *non-restrictive*. Non-restrictive clauses cannot use the word that and, when punctuated, are marked off by commas. Restrictive clauses are not marked off by commas in standard English punctuation and may use that (but are not limited to this relative pronoun), and they can in some cases be expressed without a relative pronoun. It is easiest to tell what sort of relative clause you are faced with when more than one of these differences is exhibited. For example, the relative clause The cat that the dog had chased was asleep or The cat the dog had chased was asleep is clearly restrictive while the one in The cat, who the dog had chased, was asleep is clearly non-restrictive. This means that the relative clause in the first is part of the definite description the cat that the dog had chased. The relative clause in the second would instead be analyzed as a separate conjunct to give the dog had chased the cat ∧ the cat was asleep as the initial step of the analysis.

Another indication of the difference between the two sorts or relative clause is that a non-restrictive clause can modify a proper name—as in Puff, who the dog had chased, was asleep. And, since neither prepositional phrases nor restrictive relative clauses can modify a proper name, putting a proper name in a blank that was left when you removed an apparent individual term can show whether you really removed the whole of the term. For example, Puff on the mat was asleep and Puff that the dog had chased was asleep are both ungrammatical.

Glen Helman 03 Aug 2010

6.1.s. Summary

1 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.

2 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.

3 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 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.

4 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.

5 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 or θ fits (series) $\tau_1, \ldots,$ ən τ_n.

6 In addition to proper names, the individual terms include definite descriptions and various non-anaphoric pronouns. They do not include certain other noun phrases, quantifier phrases in particular. We will speak of the "person, place, thing, or idea" referred to by an individual term by using such words as object, entity, individual, and thing, understanding these to apply to anything that might be named. Common nouns are also not individual terms. Indeed, they may be labeled general terms to distinguish their function of indicating a class of objects from the function of individual terms, also called singular terms, which is to refer to a single individual in a definite way. The word term will often be used as shorthand for individual term.

7 A functor is an operation that takes one or more individual terms as input and yields an individual term as output. Just like other operations, it expresses a function, in this case a reference function, which yields reference values when applied to reference values. Although a reference function is a particular sort of function, so the latter term is more general, we will use it term primarily for reference functions. The operation of combining a functor with input is application, and the individual term that is the output is a compound term, for which we use the symbolic notation $\zeta\tau_1\ldots\tau_n$ and the English notation ζ of τ or ζ of (series) $\tau_1, \ldots,$ ən τ_n. (The phrase applied to is sometimes a more convenient alternative to of.) For any functor, there will almost always be some terms for which the application of the functor yields an undefined term. Although this problem can be reduced by limiting the input of functors to objects of certain types, we will not include this complication in our account of logical forms.

8 It can be difficult to recognize the individual terms that fill the places of a predicate or a functor. It is important in include in a definite description all the modifiers that are part of it. Some of these may be prepositional phrases or relative clauses which follow the common noun. In some cases, a prepositional phrase in this position might either be part of a definite description or modify a verb; but such an ambiguity cannot arise with relative clauses so a prepositional phrase can be made into a relative clause in order to test what it modifies. Relative clauses must therefore be part of the definite description when they are restrictive; on the other hand, non-restrictive clauses (the sort set off by commas) are analyzed using conjunction.

Glen Helman 03 Aug 2010

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.

i. Reagan's vice president was the 41st president.

j. Tom found a fly in his soup and he called the waiter.

k. Tom found the book everyone had talked to him about and he bought a copy of it.

l. Wabash College is located in Crawfordsville, which is the seat of Montgomery County.

m. Sue and Tom set the date of their wedding but didn't decide on its location.

2. Synthesize idiomatic English sentences that express the propositions associated with the logical forms below by the intensional interpretations that follow them.

a. Wci ∧ Scl
S: [_ is south of _]; W: [_ is west of _]; c: Crawfordsville; i: Indianapolis; l: Lafayette

b. Mab → Mba
M: [_ has met _]; a: Ann; b: Bill

c. Iacb ∧ Iadb
I: [_ introduced _ to _]; a: Alice; b: Boris; c: Clarice; d: Doris

d. Wab ∧ Kabab
K: [_ asked _ to write _ about _]; W: [_ wrote to _];
a: Alice; b: Boris

e. g = c → (f = s ∧ p = t)
c: the city; f: football; g: Green Bay; p: the Packers; s: the sport; t: the team

f. (Sab ∧ ¬ Sa(fc)) → ¬ b = fc
S: [_ has spoken to _]; f: [_'s father]; a: Ann; b: Bill; c: Carol

g. (B(fa)(mb) ∨ S(ma)(fb)) → Cab
B: [_ is a brother of _]; C: [_ and _ are cross-cousins];
S: [_ is a sister of _]; f: [_'s father]; m: [_'s mother]; a: Ann; b: Bill

h. Pab(m(sb)(sc)) ∧ Pac(m(sb)(sc))
P: [_ persuaded _ to accept _]; m: [the best compromise between _ and _]; s: [_ 's proposal]; a: Ann; b: Bill; c: Carol

6.1.xa. Exercise answers

1. **a.** Ann introduced Bill to Carol

[_ introduced _ to _] Ann Bill Carol

Iabc

I fits a, b, ən c

I: [_ introduced _ to _]; a: Ann; b: Bill; c: Carol

b. Ann gave the book to either Bill or Carol

Ann gave the book to Bill ∨ Ann gave the book to Carol

[_ gave _ to _] Ann the book Bill ∨ [_ gave _ to _] Ann the book Carol

Gakb ∨ Gakc

either G fits a, k, ən b or G fits a, k, ən c

G: [_ gave _ to _]; 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 ∧ Bill gave the book to Carol

[_ gave _ to _] Ann the book Bill ∧ [_ gave _ to _] Bill the book Carol

Gakb ∧ Gbkc

both G fits a, k, ən b and G fits b, k, ən c

G: [_ gave _ to _]; 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 ∧ the package was returned to Tom

[_ had _ sent to _] Tom the package Sue ∧ [_ was returned to _] the package Tom

Htps ∧ Rpt

both H fits t, p, ən s and R fits p ən t

H: [_ had _ sent to _]; R: [_ was returned to _]; p: the package; s: Sue; t: Tom

e. Georgia will see Ed if she gets to Denver before Saturday

Georgia will see Ed ← Georgia will get to Denver before Saturday

[_ will see _] Georgia Ed ← [_ will get to _ before _] Georgia Denver Saturday

Sge ← Ggds

Ggds → Sge

if G fits g, d, ən s then S fits g ən e

G: [_ will get to _ before _]; S: [_ will see _]; 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 → I'm Sherlock Holmes

(the murderer is the butler ∨ the murderer is the nephew) → I = Sherlock Holmes

(the murderer = the butler ∨ the murderer = the nephew) → i = s

(m = b ∨ m = n) → 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

¬ (Ann saw Tom speak to either Mike or Nancy ∨ Bill saw Tom speak to either Mike or Nancy)

¬ ((Ann saw Tom speak to Mike ∨ Ann saw Tom speak to Nancy) ∨ (Bill saw Tom speak to Mike ∨ Bill saw Tom speak to Nancy))

¬ (([_ saw _ speak to _] Ann Tom Mike ∨ [_ saw _ speak to _] Ann Tom Nancy) ∨ ([_ saw _ speak to _] Bill Tom Mike ∨ [_ saw _ speak to _] Bill Tom Nancy))

¬ ((Satm ∨ Satn) ∨ (Sbtm ∨ Sbtn))

not either either S fits a, t, ən m or S fits a,t, ən n or either S fits b,t, ən m or S fits b,t, ən n

S: [_ saw _ speak to _]; 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 ← each of Ann, Bill, and Carol will ask Tom

Tom will agree ← ((Ann will ask Tom ∧ Bill will ask Tom) ∧ Carol will ask Tom)

[_ will agree] Tom ← (([_ will ask _] Ann Tom ∧ [_ will ask _] Bill Tom) ∧ [_ will ask _] Carol Tom)

Gt ← ((Aat ∧ Abt) ∧ Act)

((Aat ∧ Abt) ∧ Act) → Gt

if both both A fits a ən t and A fits b ən t and A fits c ən t then G fits t

A: [_ will ask _]; G: [_ 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 [_, _, and _ will ask _], which is what would be required in order to express instead the idea of Ann, Bill, and Carol making the request as a group.

i. Reagan's vice president was the 41st president.

Reagan's vice president = the 41st president

[_'s vice president] Reagan = [the _th president] 41

vr = pf

v of r is p of f

p: [the _th president]; v: [_ 's vice president]; f: 41; r: Reagan

j. Tom found a fly in his soup and he called the waiter

Tom found a fly in his soup ∧ Tom called the waiter

Tom found a fly in Tom's soup ∧ Tom called the waiter

[_ found a fly in _] Tom Tom's soup ∧ [_ called _] Tom the waiter

Ft(Tom's soup) ∧ Ctr

Ft([_'s soup] Tom) ∧ Ctr

Ft(st) ∧ Ctr

both F fits t ən s of t and C fits t ən r

C: [_ called _]; F: [_ found a fly in _]; s: [_'s soup]; r: the waiter; t: Tom

k. Tom found the book everyone had talked to him about and he bought a copy of it

Tom found the book everyone had talked to him about ∧ Tom bought a copy of the book everyone had talked to him about

Tom found the book everyone had talked to Tom about ∧ Tom bought a copy of the book everyone had talked to Tom about

[_ found _] Tom the book everyone had talked to Tom about ∧ [_ bought a copy of _] Tom the book everyone had talked to Tom about

Ft(the book everyone had talked to Tom about) ∧ Bt(the book everyone had talked to Tom about)

Ft([the book everyone had talked to _ about] Tom) ∧ Bt([the book everyone had talked to _ about] Tom)

Ft(bt) ∧ Bt(bt)

both F fits t ən b of t and B fits t ən b of t

B: [_ bought a copy of _]; F: [_ found _]; b: [the book everyone had talked to _ about]; t: Tom

l. Wabash College is located in Crawfordsville, which is the seat of Montgomery County

Wabash College is located in Crawfordsville ∧ Crawfordsville is the seat of Montgomery County

[_ is located in _] Wabash College Crawfordsville ∧ Crawfordsville = the seat of Montgomery County

Lbc ∧ c = [the seat of _] Montgomery County

Lbc ∧ c = sm

both L fits b ən c and c is s of m

L: [_ is located in _]; s: [the seat of _]; b: Wabash; c: Crawfordsville; m: Montgomery County

m. Sue and Tom set the date of their wedding but didn't decide on its location

Sue and Tom set the date of their wedding ∧ Sue and Tom didn't decide on the location of their wedding

Sue and Tom set the date of Sue and Tom's wedding ∧ ¬ Sue and Tom decided on the location of Sue and Tom's wedding

[_ and _ set _] Sue Tom the date of Sue and Tom's wedding ∧ ¬ [_ and _ decided on _] Sue Tom the location of Sue and Tom's wedding

Sst(the date of Sue and Tom's wedding) ∧ ¬ Dst(the location of Sue and Tom's wedding)

Sst([the date of _] Sue and Tom's wedding) ∧ ¬ Dst([the location of _] Sue and Tom's wedding)

Sst(d(Sue and Tom's wedding)) ∧ ¬ Dst(l(Sue and Tom's wedding))

Sst(d([_ and _'s wedding] Sue Tom)) ∧ ¬ Dst(l([_ and _'s wedding] Sue Tom))

Sst(d(wst)) ∧ ¬ Dst(l(wst))

both S fits s, t, ən d of (w of s ən t) and not D fits s, t, ən l of (w of s ən t)

D: [_ and _ decided on _]; S: [_ and _ set _]; d: [the date of _]; l: [the location of _]; w: [_ and _ 's wedding]; s: Sue; t: Tom

2. a. [_ is west of _] Crawfordsville Indianapolis ∧ [_ is south of _] Crawfordsville Lafayette

Crawfordsville is west of Indianapolis ∧ Crawfordsville is south of Lafayette

Crawfordsville is west of Indianapolis and south of Lafayette

b. [_ has met _] Ann Bill → [_ has met _] Bill Ann

Ann has met Bill → Bill has met Ann

If Ann has met Bill then he has met her

c. [_ introduced _ to _] Alice Clarice Boris ∧ [_ introduced _ to _] Alice Doris Boris

Alice introduced Clarice to Boris ∧ Alice introduced Doris to Boris

Alice introduced Clarice and Doris to Boris

d. [_ wrote to _] Alice Boris ∧ [_ asked _ to write _ about _] Alice Boris Alice Boris

Alice wrote to Boris ∧ Alice asked Boris to write Alice about Boris

Alice wrote to Boris ∧ Alice asked Boris to write her about himself

Alice wrote to Boris and asked him to write her about himself

e. g = c → (f = s ∧ p = t)

Green Bay = the city → (football = the sport ∧ the Packers = the team)

Green Bay is the city → (football is the sport ∧ the Packers are the team)

Green Bay is the city → 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

f. ([_ has spoken to _] Ann Bill ∧ ¬ [_ has spoken to _] Ann ([_'s father] Carol)) → ¬ Bill = [_'s father] Carol

(Ann has spoken to Bill ∧ ¬ [_ has spoken to _] Ann Carol's father) → ¬ Bill = Carol's father

(Ann has spoken to Bill ∧ ¬ Ann has spoken to Carol's father) → ¬ Bill is Carol's father

(Ann has spoken to Bill ∧ Ann hasn't spoken to Carol's father) → Bill isn't Carol's father

Ann has spoken to Bill but not to Carol's father → Bill isn't Carol's father

If Ann has spoken to Bill but not to Carol's father, then Bill isn't Carol's father

g. (B([_'s father] Ann)([_'s mother] Bill) ∨ S([_'s mother] Ann)([_'s father] Bill)) → [_ and _ are cross-cousins] Ann Bill

([_ is a brother of _] Ann's father Bill's mother ∨ [_ is a sister of _] Ann's mother Bill's father) → Ann and Bill are cross-cousins

(Ann's father is a brother of Bill's mother ∨ Ann's mother is a sister of Bill's father) → Ann and Bill are cross-cousins

Ann's father is a brother of Bill's mother or Ann's mother is a sister of Bill's father → Ann and Bill are cross-cousins

If Ann's father is a brother of Bill's mother or Ann's mother is a sister of Bill's father, then Ann and Bill are cross-cousins

h. Pab(m([_'s proposal] Bill)([_'s proposal] Carol))
∧ Pac(m([_'s proposal] Bill)([_'s proposal] Carol))

Pab([the best compromise between _ and _] Bill's proposal Carol's proposal)
∧ Pac([the best compromise between _ and _] Bill's proposal Carol's proposal)

[_ persuaded _ to accept _] Ann Bill the best compromise between Bill's proposal and Carol's proposal
∧ [_ persuaded _ to accept _] Ann Carol the best compromise between Bill's proposal and Carol's proposal

Ann persuaded Bill to accept the best compromise between his and Carol's proposals
∧ Ann persuaded Carol to accept the best compromise between Bill's proposal and hers

Ann persuaded each of Bill and Carol to accept the best compromise between their proposals

Glen Helman 03 Aug 2010

6.2. Predicates and pronouns

6.2.0. Overview

Glen Helman 03 Aug 2010

6.2.1. Abstracts

In our analyses so far, we have identified the places of a predicate with the blanks remaining when all largest individual terms have been removed. But, while this way of identifying the places of a predicate is best for a full analysis, it is not required by the concept of a predicate. For the greatest flexibility in identifying predicates, we need a notation that will allow us to specify an order for the places of a predicate that is different from the order of blanks in the English 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.

The usual form of definition for a function—of a polynomial, 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 some sort of formula. An alternative notation represents the input and output more graphically

$$f\colon x, y \mapsto x^2 + 3xy + 1$$

The latter 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 sort of definite description that appears in the English version of the definition. Using this notation, the symbolic definition could be written as

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

That is, the expression "$\lambda xy\ (x^2 + 3xy + 1)$" can be read as the function which, when given input x and y, yields the output x^2 + 3xy + 1.

When we define a function by a formula, whether we use the traditional notation or Church's, we are interested in the way the meaning of the formula varies with changes in the reference of certain individual terms. This "way" is more abstract than any particular value the formula has when the reference of these terms is fixed, so the move from the formula to the function is reasonably described as "abstraction." The notation of lambda abstraction identifies a function without immediately 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 above would now often be described as an “anonymous function.” So, when a defining equation is expressed in the notation of lambda abstraction, it abstracts a function anonymously by using the expression “$\lambda xy\ (x^2 + 3xy + 1)$” and then assigns it the name “f.”

Since predicates express functions, the same idea can be applied to them, and it will provide the sort of flexibility we need in identifying predicates. However, our notation for abstraction will be a little different from Church’s. We will write the variables that follow the lambda in Church’s notation as subscripts on brackets. For example, for the function defined above, we can write

$$[x^2 + 3xy + 1]_{xy}$$

something that might be read as $x^2 + 3xy + 1$ as a function of x and y.

As an example of a predicate in this notation, consider the following:

$$[x\ \text{introduced}\ x\ \text{to}\ y]_{xy}$$

If we give this the input Bill and Ann, it will generate an output sentence by putting Bill in place of x and Ann in place of y. The output will then have Bill in the first and second blanks of the sentence-with-blanks _ introduced _ to _, and it will have Ann in the third blank. So we will get as output the sentence Bill introduced Bill to Ann—or, more idiomatically, Bill introduced himself to Ann.

That is, the expression,

$$[x\ \text{introduced}\ x\ \text{to}\ y]_{xy}\ \underline{\text{Bill}}\ \underline{\text{Ann}}$$

provides an alternative analysis of the first example of 6.1.5 in which use a two-place predicate instead the three-place predice [_ introduced _ to _]. The chief application of this sort of flexibility in analysis will be in later chapters; but this example shows that it captures some aspects of English predications better than the analysis we will most often use. In particular, like the English sentence, this analysis indicates a double reference to Bill without repeating his name. We will look at this aspect of abstraction further in 6.2.3.

We will call an expression formed with these subscripted brackets an *abstract*. We will speak of a *predicate abstract* when the brackets enclose a sentence-with-variables and of a *functor abstract* when they enclose an individual-term-with-blanks. The general form of an abstract with *n* places is

$$[\underbrace{\ ---\ }_{\textit{body}}]_{\underbrace{x_1 \ldots x_n}_{\textit{abstractor}}}$$

It has two parts, a *body*, which specifies the output of the expression, and an *abstractor*, consisting of the brackets and subcripted list of variables. The variables listed in the abstractor may appear in the body in any order and may occur several times.

And they need not occur in the body at all. To get the effect of a definition like f(x) = 2, we use an abtract like $[2]_x$ to indicate a function whose output is 2 for any input. Abraction like this is said to be *vacuous*.

The predicate abstract $[x\ \text{introduced}\ x\ \text{to}\ y]_{xy}$ might be read as

what “x introduced x to y” says about x and y

and we will take as our English notation for predicate abstracts an abbreviated form of this reading:

what --- says of $x_1 \ldots x_n$

so the English notation for this predicate would be

what x introduced x to y says of x ən y

(again using the contraction ən to distinguish this use of and from its use in conjunction). The predication that applies this predicate to Ann and Bill then takes the form

what x introduced x to y says of x ən y fits Bill ən Ann

Our English notation for functor abstracts is simply

--- for $x_1 \ldots x_n$

which is a compact version of a reading suggested earlier. The application of $[x^2 + 3xy + 1]_{xy}\ \underline{2}\ \underline{3}$ could be written in this notation as

$x^2 + 3xy + 1$ for x ən y applied to 2 ən 3

This is a case where the alternative English notation applied to for compound functors reads better than the simpler of.

Glen Helman 03 Aug 2010

6.2.2. Bound variables

If a variable in an abstractor appears in the body of an abstract, its occurrences in the body are said to be *bound* to the abstractor. So any occurrence of $x_1, \ldots, x_n$ in the body --- of the following abstract is bound to the abstractor $x_1 \ldots x_n$:

$$[\text{ --- }]_{x_1 \ldots x_n}$$

If a variable is in the scope of more than one abstractor containing it, it is bound to the one with narrowest scope. So the first occurrence of x and the occurrence of y are bound to the abstractor xy in the following while the other occurrences of x (outside the abstractors) and the occurrence of z are bound to the abstractor xz:

$$[[y \text{ introduced } x \text{ to } z]_{xy}\ xx]_{xz}$$

A variable that is not bound to any abstractor is said to be *free*. So z is free in $[y \text{ introduced } x \text{ to } z]_{xy}$, and when an expression like “$x^2 + 3xy + 1$” or “x introduced x to y” is considered by itself outside the context of an abstract, all variables in it are free.

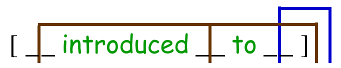
Variables have the grammatical status of individual terms but have no definite reference values. In the context of a formula like “$x^2 + 3xy + 1$,” free variables are naturally thought of as variable quantities (hence their name) since, when they vary in their reference, the value of the formula varies as a result. When variables are bound in an abstract like $[x^2 + 3xy + 1]_{xy}$, there is no longer this sort of variation. The abstract makes a reference to a mathematical object, a polynomial function, that incorporates the variation but does not itself vary. Because of this, an older terminology referred to bound variables as “apparent” variables.

The notation for predicates and functors used in 6.1 can be thought of as a variation on the notation for abstracts that deals with the “apparent” character of bound variables by removing them entirely. We will understand a bracketed sentence- or individual-term-with-blanks to represent an abstract in which each of the blanks is filled with a different variable and the variables appear in the same order in the body and the abstactor. So [_ introduced _ to _] would come to the same thing as the abstract

$$[x \text{ introduced } y \text{ to } z]_{xyz}$$

Because the blanks in the English expression correspond one for one and in the same order to the places of the predicate or functor, there is no need for bound variables to indicate the relation between the two.

Bracketing alone is not sufficient in cases where the places of a predicate do not correspond one for one to the blanks. However, we might supplement it by lines showing how places correspond to blanks.



This is clearer than the corresponding use of bound variables

$$[x \text{ introduced } x \text{ to } y]_{xy}$$

but it is significantly less convenient. Still, it is worth bearing in mind, even when bound variables are used, since the lines in the graphical notation depict the pattern of binding of variables by the abstractor.

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

$$[x \text{ introduced } x \text{ to } y]_{xy}$$
$$[y \text{ introduced } y \text{ to } z]_{yz}$$

Each says that for any input terms τ and υ (in that order), the output sentence should be τ introduced τ to υ, and pattern of binding in each would be depicted in the same way in the graphical notation.

Expressions, like these, that use different variables to indicate the same correspondence between blanks in the body and places for input will be referred to as *alphabetic variants*. Notice that alphabetic variants can use a given variable in different ways. For example, although the variable y appears in both of the abstracts above, it would be replaced by a different one of the input terms in each case.

The body of a predicate abstract is grammatically like a sentence even though it may contain free variables. It is standard to speak of an expression as *closed* if any variables it contains are bound within it and call an expression *open* if one or more of its variables is free. Logicians typically use the term *formula* for any expression that is grammatically like a sentence whether it is open or closed, and reserve the term *sentence* for closed formulas. 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.

The distinction between open and closed expressions applies to term-like expressions also, but the terminology is handled differently. Both open and closed expressions are classified as *(individual) terms* with closed expressions distinguished simply as *closed terms*.

It is time to update our notion of atomic sentences or, more generally, *atomic formulas*. 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 any sentences that go unanalyzed will still count as atomic). We will now also count as atomic any predication. Predications are compound and can even have formulas as components (albeit not immediate components), but the role of predications in derivations is sufficiently analogous to that of unanalyzed sentences for it to make sense to put them both in the same category. This analogy lies behind our use of capital letters for predicates, and it can be built 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.

6.2.3. Variables and pronouns

English has devices which function like bound variables. The force of the abstract

[_ introduced _ to _]

or the equivalent

$[x_1$ introduced x_2 to $x_3]_{x_1x_2x_3}$

can be captured in English by the expression

what is said about three people by saying that (the first introduced the second to the third)

which uses expressions like the first, the second, and so on, instead of subscripted variables. (Parentheses were used in the English displayed above simply to mark the portion corresponding to the body of the abstract.) No particular group of people is in question here, and the expressions the first, etc., do not refer to anything outside the sentence. Instead, these expressions function here much like pronouns that have three people as their antecedant. The word order differs from that used in the English notation for abstracts, but that was done merely to put the phrase “three people” before the “pronouns” that refer to it.

In the case of a one-place predicate abstract, the corresponding English can be stated with a genuine pronoun:

$[$Tom bought $x]_x$
what is said about a thing when it is said that (Tom bought it)

The blank that is marked by x in the body of the symbolic abstract is filled in the English with the pronoun it, which 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 and cannot be indefinite in independent ways. The general moral is that the variables used in the bodies of abstracts are like pronouns, and the ones in abstractors are like their antecedents. One consequence of this reiterates a point made in the last subsection: you should not expect variables bound to different abstractors to be linked in their reference any more than you would expect this of pronouns that have different antecedents.

We can also move in the other direction and use abstracts to represent the contribution of pronouns to the logical form of a sentence. We can get a hint of how they might do this by looking at a particular English rendering of the sample predicate abstract discussed in 6.2.1

$$[x \text{ introduced } x \text{ to } y]_{xy}$$

what is said about two people when it is said that (the former introduced the former to the latter)

where we use another common pronoun-like device. Now consider the following restatement:

what is said about two people when it is said that (the former introduced him- or herself to the latter)

The reflexive pronoun in this expression corresponds to the repeated variable in the symbolic abstract.

Of course this English expression was a rather artificial one constructed to correspond to an abstract, but there are ways to apply abstracts more broadly. To see how, let us look at three further English expressions corresponding to the abstracts we have been considering. This time the English expressions are predicates (rather than noun phrases that refer to the contents of predicates):

___, ___, and ___ are such that (the first introduced the second to the third)

___ is such that (Tom bought it)

___ and ___ are such that (the former introduced the former to the latter)

Of course, these predicates themselves are also artificial, but they employ a device, the various forms of the phrase is such that, that is sometimes unavoidable. And, while there are usually better ways of saying what may be said using it, it can be easily understood and may be applied to virtually any English sentence to restate it (in English) in a way that corresponds to the use of an abstract.

Because the first element of a sentence often indicates the topic under discussion, languages have many devices for restating sentences with various elements at the front. One common device in English is the use of passive voice. If we wish to say who wrote a book but focus attention on the book rather than its author, we might say something like *Moby Dick* was written by Melville. Here we take the direct object of Melville wrote *Moby Dick* and move it to the front of the sentence by changing the verb from active to passive voice. Passive voice can be used similarly to move more than direct objects to the front, but it has limitations, as do many of the other devices English has for making noun phrases into subjects. The use of is such that—which we will call *expansion*—enables us to make a great variety and arbitrary number of noun phrases into the subject of a sentence. This phrase is written after the subject and is itself followed by the result of replacing the noun phrases in the original sentence by pronouns or pronoun-like devices. For example, Melville wrote *Moby Dick* can be converted into any of

Moby Dick is such that (Melville wrote it)
Melville is such that (he wrote *Moby Dick*)
Melville and *Moby Dick* are such that (the former wrote the latter)

The result of expansion is an *expanded form*, and we will often write it, as has been done here, with the residue of the original sentence in parentheses. When we need to distinguish among alternative ways of expanding a sentence, we will speak of expanding on a particular noun phrase. The opposite of expansion is *reduction*, and we will describe the original sentence as being in *reduced form* relative to that expansion. The idea of reduced form is relative because, in principle, expansion can be applied more than once, and a reduced form may be reduced still further. For example, the first expanded form above is also the result of reducing *Moby Dick* is such that (Melville is such that (he wrote it)).

Expansion will serve us in a number of different ways in the rest of the course. For now, the fact that it uses pronouns and is analogous to the use of abstracts will help in using abstracts to analyze the role of pronouns in a sentence. To see how, let us analyze the sentence Bill told Ann his name in a way that employs a predicate abstract to reflect the use of a pronoun.

Bill told Ann his name

$\underline{\text{Bill}}$ is such that (he told Ann his name)

$$[\ \underline{x}\ \text{told}\ \underline{\text{Ann}}\ \underline{x\text{'s name}}\ ]_x\ \underline{\text{Bill}}$$

$$[\,[\ \_\ \text{told}\ \_\ \_\ ]\ \underline{x}\ \underline{\text{Ann}}\ \underline{x\text{'s name}}\ ]_x\ \underline{\text{Bill}}$$

$$[\text{Txa}([\ \_\text{'s name}]x)]_x\ \underline{\text{Bill}}$$

$$[\text{Txa}(\text{nx})]_x\text{b}$$

T: [_ told _ _]; n: [_'s name]; a: Ann; b: Bill

Once the sentence as a whole has been analyzed as the predication of an abstract, the formula x told Ann x's name that is the body of the abstract is analyzed in the same way as Bill told Ann Bill's name would be. The final analysis departs from the original sentence in having the equivalents of two pronouns instead of one (as does Bill is such that (he hold Ann his name), but it is like the original in having only a single occurrence of Bill. So, in this respect, it is closer to the English than the alternative analysis as Tna(nb), which is what we would get if we analyzed the sentence, Bill told Ann Bill's name, that is the result of replacing the pronoun his by its antecedent. It is in this way that expansion and analysis by abstracts reflects the use of pronouns.

We might also have expanded on both Bill and Ann to get Bill and Ann are such that he told her his name, with the analysis

$$[\mathrm{Txy(nx)}]_{xy}\mathrm{ba}$$

That would have added no enlightenment in the case of this sentence, but consider the following ambiguous sentence, given with abbreviated analyses of two interpretations of it. (Imagine that the second concerns a case of amnesia.)

Bill told Al his name	Bill told Al his name
Bill and Al are such that (the former told the latter the former's name)	Bill and Al are such that (the former told the latter the latter's name)
[x told y x's name]$_{xy}$ Bill Al	[x told y y's name]$_{xy}$ Bill Al
$[\mathrm{Txy(nx)}]_{xy}\mathrm{bl}$	$[\mathrm{Txy(ny)}]_{xy}\mathrm{bl}$

T: [_ told _ _]; n: [_'s name]; b: Bill; l: Al

In each of these analyses, the names Bill and Al are separated completely from the abstracts, which use variables to show any patterns of coreference. The advantage of this sort of analysis is that it gives us an account of the ambiguity of this sentence that enables us to point to the same ambiguity in other sentences, such as Barb told Ann her name.

6.2.4. Expanded and reduced forms

We will use the ideas of expansion and reduction and of expanded and reduced form in connection with symbolic analyses as well as English sentences. (The context will usually indicate which use of the terms is intended but, if necessary, we can speak of symbolic expansion on the one hand or expansion using such that on the other.) As it applies to symbolic analyses, expansion is the process of restating an analysis using an abstract as we did when we moved from the analysis Tbab of Bill told Ann his name to $[\mathrm{Txa(nx)}]_{x}\mathrm{b}$.

From one point of view, there is no need to use expansion to study the ambiguity of Bill told Al his name. A pair of simple reduced forms like Tba(nb) and Tba(na) is quite sufficient. And even the point that Barb told Ann her name shares the same ambiguity can be captured by referring to a pair of logical forms $\mathrm{T}\tau\upsilon(\mathrm{n}\tau)$ and $\mathrm{T}\tau\upsilon(\mathrm{n}\upsilon)$ that are exhibited by each of the two pairs.

Of course, this simpler approach would ignore the fact that the ambiguity lies in the pattern of coreference marked by anaphoric pronouns. Abstracts capture this, but in a rather crude way since they introduce extra pronouns to do so. While the English sentence Bill told Al his name has a single pronoun, our analyses each had three bound variables. The notation could be modified to be more subtle if our main interest was in anaphoric pronouns with individual terms as antecedents. However, the prime application of abstracts will be in later chapters where we will use abstracts in connection with our analysis of quantifier phrases.

In order to analyze a sentence as a truth-functional compound, we must be able to identify components that function independently. In particular, a pronoun in one component cannot have its antecedent in another. The approach we took before employing abstracts was to simply replace a pronoun by its antecedent when this was possible and avoid analysis when it was not. The prime example of a pronoun we could not replace is one whose antecedent is a quantifier phrase. The sort of analysis we will eventually use in this case employs abstracts behind the scenes, and the use of abstracts for cases where pronouns have individual terms as antecedents brings those cases closer to our handling of cases where the antecedents are quantifier phrases.

Still, one of the key points to be made about abstracts with regard to individual terms is the very fact that they are dispensible, so let us look more closely at how to dispense with them once we have used them in an analysis. For example, consider the sentence Ann visited the class and she spoke to Davie. If we use an abstract to capture the coreference of she and Ann, we can analyze this as follows:

Ann visited the class and she spoke to Davie

Ann is such that (she visited the class and she spoke to Davie)

[_ is such that (she visited the class and she spoke to Davie)] Ann

[x visited the class and x spoke to Davie]$_x$ Ann

[x visited the class ∧ x spoke to Davie]$_x$ Ann

[[_ visited _] x the class ∧ [_ spoke to _] x Davie]$_x$ Ann

$[Vxc \land Sxd]_x a$

what both Vxc and Sxd says of x fits a

S: [_ spoke to _]; V: [_ visited _]; a: Ann; c: the class; d: Davie

The formula x visited the class and x spoke to Davie can be analyzed as a truth-functional compound because the two occurences of the variable x are independent of each other (though each is bound to the abstractor).

The approach we used earlier would have led us to analyze the sentence as the compound Ann visited the class ∧ Ann spoke to Davie in which she is replaced by Ann, and this sentence would receive a symbolic analysis of the form Vac ∧ Sad. Now, if we compare the symbolic analyses

$$[Vxc \land Sxd]_x a \qquad Vac \land Sad$$

we can see that the second is the result of putting the term a in place of the variable x in the body of the abstract in the first. That is, the second is the reduced form of the first.

When we reduce the predication of an abstract, we take the body of the abstract and put the term of which it is predicated in the blanks marked by the variable. An analogous description applies to the reduction of compound terms formed by applying functor abstracts, and the description can be extended to apply to abstracts on any number of variables. Schematically, the general pattern is as follows:

$$[\text{---}x_1\text{---}\ldots\text{---}x_n\text{---}]_{x_1\ldots x_n}\tau_1\ldots\tau_n \qquad \text{---}\tau_1\text{---}\ldots\text{---}\tau_n\text{---}$$

When interpreting the schema, remember that the variables of the abstractor can appear in the body in any order and may each appear any number of times (including not at all). The expression on the right is the result of using each term τ_i to replace all occurrences of the corresponding variable.

Special care is needed when performing a reduction if the body contains abstracts and a term to be substituted contains free variables. The short account of this sort of case is that no free variable should become bound as a result of reduction and that abstracts should be replaced by alphabetic variants as necessary to avoid this happening. The easiest way to insure this is to choose bound variables so that they are all different from each other and from any free variables. However, our use of abstracts will be limited to much simpler situations, so a detailed rule is not important. Moreover, we will regard reduced and expanded expressions as two ways of writing the same formula or term, so no rule at all is needed as part of our rules for derivations, where sentences will be written only in fully reduced form.

Let us now return to the issue of pronouns and truth-functional connectives. From our present point of view, the fact that pronouns can always be replaced by individual term antecedents can be seen as the result of the fact the reduction is always possible. The analyses of sentences involving quantifier phrases that we will go on to develop in the next couple of chapters will employ predicate abstracts but not by way of predication, so nothing analogous to reduction will be in question. That can be cited as the reason a pronoun often cannot be replaced by a quantifier phrase antecedent—as in A mother visited the class and she spoke to Davie, which is not equivalent to A mother visited the class and a mother spoke to Davie. In cases where replacement by a quantifier phrase antencedent is possible without changing the meaning—as in A mother visited the class or she spoke to Davie on the phone—this will be due to special interactions between the quantifier phrase and other logical constants in the sentence.

Finally, although our focus has been on pronouns, much of what we have seen applies also to sentences containing compound predicates and other compound phrases. The sentence Ann visited the class and spoke to Davie can also be analyzed as $[Vxc \land Sxd]_x a$. While this analysis introduces the symbolic analogues pronouns that do not appear in the English, it does capture the form of the English in one respect: it treats it as a predication whose predicate contains the connective. And the possibility of restating the sentence as Ann visited the class and Ann spoke to Davie can be seen as due to the reduction of this form to Vac ∧ Sad.

6.2.s. Summary

1 We adapt the notation of lambda abstraction to provide a flexible way of linking the places of a predicate to blanks in an English sentence. An expression formed using our notation—which will have the general form $[\ldots x_1 \ldots x_n \ldots]_{x_1 \ldots x_n}$—is an abstract (in this use, a predicate abstract); it consists of a abstractor applied to a parenthesized body. In English notation, a predicate abstract takes the form what … x_1… x_n … says of x_1 … x_n, and a functor abstract takes the form … x_1… x_n … for x_1 … x_n. (Variables in an abstractor that do not appear in the body are cases of vacuous abstraction.)

2 A variable in the body of an abstract that appears in an abstractor is bound to it, provided it is not already bound to one with narrower scope. Bound variables may be thought of as pronouns whose antecedents are in the abstractor. Expressions that establish the same patterns of binding using different variables are alphabetic variants. A expression that has variables not bound to any abstractor (such as the body of an abstract considered by itself) is open; otherwise, it is closed. A sentence-like expression that is open 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.)

3 Many pronouns in English function like the bound variables of the symbolic notation for abstracts, and the phrase is such that can be used to expand an English sentence by introducing them. The resulting expanded form is analogous to the predication of an abstract and can be reduced to a sentence in which the pronouns introduced by expansion are replaced by their antecedents. Because of the analogy between variables and anaphoric pronouns, abstracts can be used to represent the contribution of such pronouns to logical form.

4 Processes analogous to the expansion and reduction of English sentences apply to symbolic forms. In the simplest case, the application of an abstract can be reduced by replacing variables bound to it by the terms filling the corresponding places of the predicates. And a symbolic form may be expanded to introduce the predication of an abstract. Both operations help in comparing sentences in reduced form to logical forms studied in later chapters in which abstracts appear in contexts other than predication.

6.2.x. Exercise questions

1. Expand each of the following in two different ways, (i) on a single occurrence of a single individual term, and (ii) on all terms together. In each case express the expanded form in English using is such that and in a partially symbolic way, as in

[x wrote *Moby Dick*]$_x$ Melville

a. Romney is north of Linden.

b. Mike gave the package to Nancy.

c. Tom spoke of Ed to Sue.

d. Sam traveled to Atlanta by way of Chicago.

2. Analyze each of the following in a way that uses abstracts and variables to represent pronouns instead of replacing them by their antecedents. Since you will not replace pronouns by their antecedents, you should end up with as many occurrences of each individual term in your result as in the original sentence. Also, restate your symbolic analysis in reduced form.

a. Ann nominated herself

b. Ralph tried the motor, and it started

c. If the alarm is touched, it will go off

d. Ralph fixed Sam's car, and he drove it back to him

e. Ann and Bill each left a message for the other

3. Each of the following sentences exhibits an ambiguity (in pronoun reference) between meanings that can be indicated by alternative analyses using abstracts. Use abstracts to give two complete analyses of each sentence that express different interpretations of it. You will find it easier to distinguish interpretations if you expand for all terms involved in the ambiguity whether or not all have pronouns referring to them on each interpretation (see the last example of 6.2.3).

In **c**, the word so serves to apply the same predicate to Bill as was applied to Al, so each of your analyses of it should have a repeated abstract.

a. Al called Bill, and he called Carol.

b. Sam gave the book to Tom, but he didn't read it.

c. Al washed his car, and so did Bill.

4. For each of the following abstracts (i) diagram the pattern of binding using lines rather than variables (in the manner shown in 6.2.2) and (ii) give an alphabetic variant (i.e., abstract which indicates the same pattern of binding using different variables).

In the case of **e**, remember that, as noted in 6.2.2, a bracketed sentence-with-blanks amounts to an abstract whose body has a different variable in each blank and whose abstractor lists the variables in the same order. Also, the lower-case f in **c** means that it is a functor rather than a predicate; but that won't make for any differences in the way you handle it.

a. $[Fx]_x$

b. $[Fz \rightarrow Gz]_z$

c. $[Tyxy]_{xy}$

d. $[fyz]_{zy}$

e. [S _ _ _]

f. $[[Rxy]_x a \wedge Rby]_y$

g. $[[Rcy]_y a \wedge Rby]_y$

Glen Helman 03 Aug 2010

6.2.xa. Exercise answers

1. In each case, the English restatement appears first, followed by the partial symbolization.

a. **i.** Romney is such that (it is north of Linden)
[x is north of Linden]$_x$ Romney

or Linden is such that (Romney is north of it)
[Romney is north of x]$_x$ Linden

ii. Romney and Linden are such that (the former is north of the latter)
[x is north of y]$_{xy}$ Romney Linden

b. **i.** Mike is such that (he gave the package to Nancy)
[x gave the package to Nancy]$_x$ Mike

or the package is such that (Mike gave it to Nancy)
[Mike gave x to Nancy]$_x$ the package

or Nancy is such that (Mike gave the package to her)
[Mike gave the package to x]$_x$ Nancy

ii. Mike, the package, and Nancy are such that (the first gave the second to the third)
[x gave y to z]$_{xyz}$ Mike the package Nancy

c. **i.** Tom is such that (he spoke of Ed to Sue)
[x spoke of Ed to Sue]$_x$ Tom

or Ed is such that (Tom spoke of him to Sue)
[Tom spoke of x to Sue]$_x$ Ed

or Sue is such that (Tom spoke of Ed to her)
[Tom spoke of Ed to x]$_x$ Sue

ii. Tom, Ed, and Sue are such that (the first spoke of the second to the third)
[x spoke of y to z]$_{xyz}$ Tom Ed Sue

d. **i.** Sam is such that (he traveled to Atlanta by way of Chicago)
[x traveled to Atlanta by way of Chicago]$_x$ Sam

or Atlanta is such that (Sam traveled to it by way of Chicago)
[Sam traveled to x by way of Chicago]$_x$ Atlanta

or Chicago is such that (Sam traveled to Atlanta by way of it)
[Sam traveled to Atlanta by way of x]$_x$ Chicago

ii. Sam, Atlanta, and Chicago are such that (the first traveled to the second by way of the third)
[x traveled to y by way of z]$_{xyz}$ Sam Atlanta Chicago

2. **a.** Ann nominated herself
Ann is such that (she nominated herself)
[x nominated x]$_x$ Ann
$[Nxx]_x a$
reduced form: Naa
N: [_ nominated _]; a: Ann

b. Ralph tried the motor, and it started
the motor is such that (Ralph tried it, and it started)
[Ralph tried x and x started]$_x$ the motor
[Ralph tried x ∧ x started]$_x$ m
$[Trx \land Sx]_x m$
reduced form: Trm ∧ Sm
S: [_ started]; T: [_ tried _]; m: the motor; r: Ralph

The analysis $[Txy \land Sy]_{xy} rm$ is also correct, but a 2-place abstract is not needed in order to analyze pronouns since only the motor has a pronoun referring to it.

c. If the alarm is touched, it will go off
the alarm is such that (if it is touched, it will go off)
[if x is touched, x will go off]$_x$ the alarm
[x will be touched → x will go off]$_x$ a
$[Tx \to Gx]_x a$
reduced form: Ta → Ga
T: [_ will be touched]; G: [_ will go off]; a: the alarm

d. Ralph fixed Sam's car, and he drove it back to him
Ralph and Sam are such that (the former fixed the latters's car, and he drove it back to him)
[x fixed y's car, and x drove y's car back to y]$_{xy}$ Ralph Sam
[x fixed y's car ∧ x drove y's car back to y]$_{xy}$rs
[Fx(y's car) ∧ Dx(y's car)y]$_{xy}$rs
$[Fx(cy) \land Dx(cy)y]_{xy} rs$
reduced form: Fr(cs) ∧ Dr(cs)s
D: [_ drove _ back to _]; F: [_ fixed _]; c: [_'s car]; r: Ralph; s: Sam

e. Ann and Bill each left a message for the other
Ann and Bill are such that (they each left a message for the other)
[x and y each left a message for the other]$_{xy}$ Ann Bill
[x left a message for y ∧ y left a message for x]$_{xy}$ab
$[Mxy \land Myx]_{xy} ab$
reduced form: Mab ∧ Mba
M: [_ left a message for _]; a: Ann; b: Bill

The noun phrase a message is a quantifier phrase rather than an individual term so it must be left unanalyzed.

3. **a.** **i.** Al called Bill, and he called Carol

Al and Bill are such that (the former called the latter, and the former called Carol)

[x called y, and x called Carol]$_{xy}$ $\underline{\text{Al}}$ $\underline{\text{Bill}}$

[x called y ∧ x called $\underline{\text{Carol}}$]$_{xy}$ab

[Cxy ∧ Cxc]$_{xy}$ab

ii. Al called Bill, and he called Carol

Al and Bill are such that (the former called the latter, and the latter called Carol)

[x called y, and y called Carol]$_{xy}$ $\underline{\text{Al}}$ $\underline{\text{Bill}}$

[x called y ∧ y called $\underline{\text{Carol}}$]$_{xy}$ab

[Cxy ∧ Cyc]$_{xy}$ab

C: [_ called _]; a: Al; b: Bill; c: Carol

The second interpretation can be indicated in spoken English by emphasizing the pronoun. The first interpretation could be indicated unambiguously by adding too to the end of the sentence.

b. **i.** Sam gave the book to Tom, but he didn't read it

Sam, the book and Tom are such that (the first gave the second to the third, but the first didn't read the second)

[x gave y to z, but x didn't read y]$_{xyz}$ $\underline{\text{Sam}}$ $\underline{\text{the book}}$ $\underline{\text{Tom}}$

[x gave y to z ∧ ¬ x read y]$_{xyz}$sbt

[Gxyz ∧ ¬ Rxy]$_{xyz}$sbt

ii. Sam gave the book to Tom, but he didn't read it

Sam, the book and Tom are such that (the first gave the second to the third, but the third didn't read the second)

[x gave y to z, but z didn't read y]$_{xyz}$ $\underline{\text{Sam}}$ $\underline{\text{the book}}$ $\underline{\text{Tom}}$

[x gave y to z ∧ ¬ z read y]$_{xyz}$sbt

[Gxyz ∧ ¬ Rzy]$_{xyz}$sbt

G: [_ gave _ to _]; R: [_ read _]; b: the book; s: Sam; t: Tom

It is hard to avoid this ambiguity in English without some rewording —e.g., by resorting to the former or the latter instead of he or by repeating one of the names.

c. **i.** Al washed his car, and so did Bill.

Al and Bill are such that (the former washed his car, and so did the latter).

[x washed his car, and so did y]$_{xy}$ $\underline{\text{Al}}$ $\underline{\text{Bill}}$

[x is such that (he washed his car) ∧ y is such that (he washed his car)]$_{xy}$ab

[[z washed $\underline{\text{z's car}}$]$_z$x ∧ [z washed $\underline{\text{z's car}}$]$_z$y]$_{xy}$ab

[[Wz(z's car)]$_z$x ∧ [Wz(z's car)]$_z$y]$_{xy}$ab

[[Wz(cz)]$_z$x ∧ [Wz(cz)]$_z$y]$_{xy}$ab

ii. Al washed his car, and so did Bill.

Al and Bill are such that (the former washed his car, and so did the latter).

[x washed his car, and so did y]$_{xy}$ $\underline{\text{Al}}$ $\underline{\text{Bill}}$

[x is such that (he washed x's car) ∧ y is such that (he washed x's car)]$_{xy}$ab

[[z washed $\underline{\text{x's car}}$]$_z$x ∧ [z washed $\underline{\text{x's car}}$]$_z$y]$_{xy}$ab

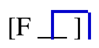
[[Wz(x's car)]$_z$x ∧ [Wz(x's car)]$_z$y]$_{xy}$ab

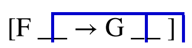
[[Wz(cx)]$_z$x ∧ [Wz(cx)]$_z$y]$_{xy}$ab

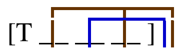
W: [_ washed _]; c: [_'s car]; a: Al; b: Bill

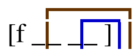
The abstracts here serve two different purposes. The one with largest scope is used to analyze the patterns of co-reference while the two inside its body are designed to capture the function of so did. The ambiguity in the sentence arises because the sameness claimed for Al's and Bill's actions might suggest washing a car related to the washer in the same way (the first interpretation) or, indeed, washing the very same car (the second interpretation). In particular, it's the difference between the idea of washing one's own car—i.e., [Wz(cz)]$_z$—and washing the car of someone, x—i.e., [Wz(cx)]$_z$—someone who, in this case, is the first person to whom the predicate is applied. It is the function of the abstract with wider scope to capture this idea of a reference to the first person to whom the predicate is applied.

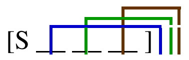
4. Since each abstract has many (indeed, infinitely many) alphabetic variants, the answers **(ii)** below are only examples.

a. **i.**  **ii.** [Fy]$_y$

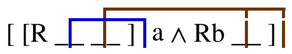
b. **i.**  **ii.** [Fx → Gx]$_x$

c. **i.**  **ii.** [Txyx]$_{yx}$

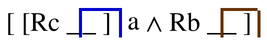
d. **i.**  **ii.** [fzx]$_{xz}$

e. **i.** 

ii. $[Sxyz]_{xyz}$

f. **i.** 

ii. $[[Ryx]_y a \land Rbx]_x$

g. **i.** 

ii. $[[Rcx]_x a \land Rbz]_z$

In the original abstract for **(g)**, $[[Rcy]_y a \land Rby]_y$, the variable y in Rcy falls in the scope of two abstractors for y. It is bound to the one with narrower scope, so the one with wider scope binds only the y in Rby. The pattern in **(i)** shows that the variable in the first abstract is thoroughly “apparent” from the point of the abstractor with wider scope: since the latter binds no variables in the first abstract, it does not matter whether that abstract uses the same variable as it does or a different one. In **(f)**, on the other hand, the two abstractors must use different variables since one binds variables in the scope of the other.

Glen Helman 19 Oct 2010

6.3. Arguments involving equations

6.3.0. Overview

The basic principles of entailment for identity are among the most familiar of logical principles; but, because equations do not have sentential components, they will play a role in derivations that is quite different from other logical forms we study.

6.3.1. Logical properties of identity

For our purposes, identity amounts to sameness in all respects, a sameness that implies interchangeability as input for any predicate or functor.

6.3.2. A law for aliases

Many of the valid conclusions from a group of equations can be captured by rules telling when terms count as “co-aliases”—i.e., aliases for the same thing.

6.3.3. Derivations for identity

The key rules for identity are rules for closing gaps, but all rules can be extended to reflect the interchangeability of co-aliases.

Glen Helman 03 Aug 2010

6.3.1. Logical properties of identity

The logical properties of identity come from two sources. One is the kind of extension we have stipulated for this relation, the pairs of reference values we say it is true of. The other is the requirement that predicates and functors be extensional, that the compounds they form be transparent to the reference values of their component terms. Properties deriving from this second source are equally properties of the operations of predication and functional application; but since they are not properties of any particular predicate or functor, it is easiest to ascribe them, along with properties of the first sort, to the logical constant =. We will turn first to the properties of identity alone.

What do we know when we know that an equation $\tau = \upsilon$ is true? Well, we know that the terms τ and υ have the same reference value; loosely speaking, we know that they name the same thing. (This is loose speech, first, because the terms may not be names but rather definite descriptions and, second, because the reference value of the terms may be nil, in which case neither names anything.) So we might say that τ and υ are each "aliases" of their common reference value in the sense that each is another name for it. It will be convenient to have a way of speaking of such terms in relation to each other rather than in relation to their value, so let us say that they are aliases in relation to each other—or, more briefly, that they are *co-aliases*.

This leads us immediately to a property of identity. For the relation of having the same reference value, of being co-aliases, is *symmetric*. It does not order the two terms in any way; if we can assert it of them taken in one order, we can equally we assert it of them the other way around. This gives us our first principle for =:

LAW OF SYMMETRY FOR IDENTITY. $\tau = \upsilon \vDash \upsilon = \tau$ (for any terms τ and υ).

This principle is stated as an entailment, but it implies that the two equations are equivalent since it licenses reversals of equations and we can undo a reversal by reversing again.

Now suppose that we know not only that a term σ is a co-alias of a term τ but also that τ is a co-alias of a term υ. All three terms must then have the same reference value, so we could say that υ is an alias for σ. Putting this more formally, we have a second principle:

LAW OF TRANSITIVITY FOR IDENTITY. $\sigma = \tau, \tau = \upsilon \vDash \sigma = \upsilon$ (for any terms σ, τ, and υ).

Again, there is a more symmetric principle lurking in the background —namely, that any two of these three equations entails the third. But this principle is harder to state compactly, and a fuller investigation of it would show that it also relies on the law of symmetry.

The two laws we have stated tell us that certain equations are true *if* others are, but they do not commit us categorically to the truth of any equations at all. How do we know there are any true equations? Well, what would it take for there to be none. Perhaps this would be so if were no aliases in the ordinary sense and every reference value was the extension of at most one term. We do not want to rule this out, for our laws are supposed to be very general and should not make any assumptions about the richness of our non-logical vocabulary. But even in a case like this, if there is any term at all in our language, we can form an equation with this term taken twice and the equation will be true. And that is one way of stating a third principle for identity:

LAW OF REFLEXIVITY FOR IDENTITY. $\vDash \tau = \tau$ (for any term τ).

So there will be true equations if there are any equations at all.

We have found three properties of identity that derive from the kind of extension we have stipulated for =. Collecting them, we have:

REFLEXIVITY. $\vDash \tau = \tau$.
SYMMETRY. $\tau = \upsilon \vDash \upsilon = \tau$.
TRANSITIVITY. $\sigma = \tau, \tau = \upsilon \vDash \sigma = \upsilon$.

Identity is not the only predicate that has these properties. For example, the predicate [_ has the same shape as _] obeys analogous laws; and that example should suggest many others. As was noted in 1.2.3, a relation for which laws of reflexivity, symmetry, and transitivity hold is said to be an *equivalence* relation.

An extreme example of an equivalence relation is the relation that holds between any pair of reference values (including any reference value and itself). Since this relation never fails to hold there is no way for it to violate any of the three laws, and it must be an equivalence relation. The extension of the identity predicate is at the other extreme of equivalence relations. If we represent the two in tabular form as truth-valued functions of reference values, we have something like this.

	0	1	2	3	⋯
0	T	T	T	T	⋯
1	T	T	T	T	⋯
2	T	T	T	T	⋯
3	T	T	T	T	⋯
⋮	⋮	⋮	⋮	⋮	⋱

=	0	1	2	3	⋯
0	T	F	F	F	⋯
1	F	T	F	F	⋯
2	F	F	T	F	⋯
3	F	F	F	T	
⋮	⋮	⋮	⋮		⋱

The first relation has **T** everywhere while the extension of = has **T** only along the diagonal from the upper left to the lower right. Identity holds in the fewest cases possible for an equivalence relation because, if any of the pairs along the diagonal were dropped, the law of reflexivity would not hold. Since identity thus expresses the narrowest equivalence relation, we might think of it as expressing sameness in all respects.

Although the status of identity as the narrowest equivalence relation derives from the extension we have stipulated for =, this does not provide a property that we can express in laws for = alone. Our ability to express the idea of sameness in all respects depends on the predicates and functors we have available to express a variety of "respects." What we can say is that identity implies sameness with regard to each predicate and functor, and we can find further properties of identity by exploiting the consequences of this idea. First consider a one-place predicate—say [_ is red]. Two things are the same with respect to redness if both are red or neither is. Hence, to say that identity implies sameness with respect to this predicate is say that an equation $\tau = \upsilon$ implies that τ is red and υ is red have the same truth value. Although we cannot express this sort of relation among three sentences directly, the symmetry of = means that it is enough to say that $\tau = \upsilon$ and τ is red together entail υ is red. Generalizing this to any one-place predicate F leads us to assert the law

$$\tau = \upsilon,\ \mathrm{F}\tau \vDash \mathrm{F}\upsilon.$$

This is as at least part of what is involved in saying that identity implies sameness in all respects. In fact, if we put θ in place of F and thus allow the predicate to be an abstract, this law says it all. But, for the moment, we will consider only predicates that are not abstracts and say that an equivalence relation that supports a law of this form for a given predicate F is a *congruence for* F. An equivalence relation that implies sameness with respect to redness (e.g., the extension of [_ has the same color as _]) is thus a congruence for [_ is red]. (The source of the term congruence is the geometrical relation of congruence, which implies sameness with respect to size and shape though not with respect to location.)

The form of this law ought to suggest something that is familiar from elementary algebra, the use of an equation to replace one expression by another. Now, in algebra, we can equally well use more than one equation to make several replacements simultaneously, and congruence principles can take a similar form. Consider sameness with respect to the relation expressed by a 2-place predicate such as [_ is younger than _]. Things that are the same in this respect should be younger than the same things and have the same things younger than them. We can express this idea compactly by the following:

$$\tau_1 = \upsilon_1,\ \tau_2 = \upsilon_2,\ \tau_1 \text{ is younger than } \tau_2 \vDash \upsilon_1 \text{ is younger than } \upsilon_2$$

And we can claim this holds for 2-place predicates generally by stating the law

$$\tau_1 = \upsilon_1,\ \tau_2 = \upsilon_2,\ \mathrm{R}\tau_1\tau_2 \vDash \mathrm{R}\upsilon_1\upsilon_2.$$

In these statements, we have economized by speaking of both places of the predicate in a single law. Since τ_1 and υ_1 could be the same term and so could τ_2 and υ_2, the law covers also cases where a change is made in only one of the two places of R. An equivalence relation that supports a law like this one for identity is said to be a *congruence for* the predicate R. The relation of having the same age will be a congruence in this sense for the relation expressed by [_ is younger than _].

Now it should be clear that we might state a law like these two that applies to identity and a predicate P with any number of places:

Congruence for P. $\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n, \mathrm{P}\tau_1\ldots\tau_n \vDash \mathrm{P}\upsilon_1\ldots\upsilon_n$ (for any terms $\tau_1, \ldots, \tau_n, \upsilon_1, \ldots, \upsilon_n$ and any predicate P with *n* places).

A large part of what we mean by saying that identity implies sameness in all respects can be captured by saying that it is a congruence for all predicates.

A large part, but not all. We have not yet said anything about functors. Here we can make the story short because the law we want is more familiar. It is this:

Congruence for f. $\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n \vDash \mathrm{f}\tau_1\ldots\tau_n = \mathrm{f}\upsilon_1\ldots\upsilon_n$ (for any terms $\tau_1, \ldots, \tau_n, \upsilon_1, \ldots, \upsilon_n$ and any functor f with *n* places).

This says that an equation between compound terms $\mathrm{f}\tau_1\ldots\tau_n$ and $\mathrm{f}\upsilon_1\ldots\upsilon_n$ follows from equations between their corresponding components. We can have laws like this for equivalence relations besides identity; and, when we have such a law for an equivalence relation, the relation is said to be a *congruence for* the functor f. The relation that holds between numbers when they have the same absolute value (i.e., of being equal or differing only in sign) is a congruence for a functor expressing the squaring function (or the cosine function). In the case of identity, we can claim congruence for *all* functors.

Have we now captured the properties of identity by saying that it is a congruence for all predicates and all functors? The laws we have stated suffice to capture all true general principles of entailment involving identity, and that was our aim. We might still ask whether a relation could obey these laws without being a relation of sameness in all respects. The question comes to something like this: are the features of a thing that are expressible by predicates and functors sufficient to pin down its identity, to distinguish it from all other things? This is a puzzling question. While any given collection of predicates and functors can certainly fail to express differences among things, it is hard to pin down the claim that there could be such differences that are expressible by no predicates or functors whatsoever, for any attempt to say what such differences might be would begin to undercut the claim that they are inexpressible. In any case, asserting the laws above for all predicates and functors suffices to establish all general principles of entailment concerning identity that we can express using our analysis of logical form.

In saying that identity is a congruence for predicates and functors, we say that predicates and functors are extensional operations and, in particular, that they form referentially transparent compounds. For example, if we were to count the sentence-with-a-gap For the past two centuries, ___ has been over 35 as a predicate, we could not say that identity is a congruence for all predicates because to assert congruence for this incomplete expression would be to assert the validity of the argument

For the past two centuries, the U. S. president has been over 35
The U. S. president = Barack Obama

For the past two centuries, Barack Obama has been over 35

and, as was noted in 6.1.3, this is naturally understood to have true premises and a false conclusion.

This raises a wider philosophical and logical issue. Could we at least say that this sentence-with-a-blank has an extension that is a function? Such a function would have to yield truth values as output based on something beyond the reference values of the terms to which it was applied, and we might speak of it as an *intensional property* (as distinct from as a *property in intension*, which is merely the way the extensional property expressed by an ordinary predicate varies from world to world). So one part of the question we have just asked is whether there are intensional properties.

The other part is whether there is anything for an intensional property to be a property of. It cannot be a property of an object thought of as a reference value because it depends on distinctions that are ignored when we say what reference value a term has. One way of putting this side of the issue is to ask whether there is any sense of *thing* in which the terms the U. S. president and Barack Obama could be said to signify different things. Perhaps we could say that one signifies a public official and the other signifies a person and say that one and the same public official could be identical with different people at different times. The oddity of this talk suggests that nasty problems might lurk here, so we will not open this door any wider. Suffice it to say that logicians and philosophers have adopted a full range of positions on this issue. Some happily accept *intensional entities* (such as public officials as distinct from the people who happen to hold offices) while others reject all talk of intensions, not only of intensional entities and intensional properties but even of the intensions of ordinary extensional predicates.

Glen Helman 03 Aug 2010

6.3.2. A law for aliases

We have seen that identity satisfies laws of reflexivity, symmetry, and transitivity and that it is a congruence for all predicates and functors. It is also possible to describe the logical properties of identity using a smaller set of fundamental laws. For example, if we include identity itself among the predicates and functors for which identity obeys laws of congruence, the reflexivity of identity will insure that it is an equivalence relation—that is, any reflexive relation that is a congruence for itself will be symmetric and transitive also. Moreover, if identity is a reflexive relation and a congruence for all one-place predicates, simple and complex (i.e., including abstracts), it is a congruence for all predicates and functors whatsoever. These simpler ways of describing the logical properties of identity are often used; but, when identifying law to implement in derivation rules, it will be most convenient to group those properties together in yet a different way.

We will have two principles, one of which will be the law asserting that identity is a congruence for all simple predicates of any number of places. The other will be a single law that groups reflexivity, symmetry, and transitivity together with the law of congruence for functors.

We arrived at the laws of symmetry and transitivity by understanding an equation $\tau = \upsilon$ to say that the two terms τ and υ are co-aliases, and we might have arrived at the law of reflexivity in the same way since in the usage of *alias* we have adopted here each term is a co-alias of itself. We can extend these same ideas more generally by speaking of terms τ and υ as being *co-aliases given* a set Δ of equations or as being *made co-aliases by* Δ. Our intention is that this relation capture the conditions under which equations are entailed by other equations.

Clearly a set Δ of equations will imply that an equation $\tau = \upsilon$ is true if either an equation between τ and υ (in either order) appears in Δ or one term can be reached from the other via a series of terms, each of which is linked to the next (in either order) by an equation in Δ. For example, if the equations shown in Figure 6.3.2-1 are in a set Δ, the terms a and e are co-aliases given Δ, as are any other pair of terms appearing in the list.

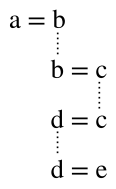


Fig. 6.3.2-1. A chain of equations making terms a and e co-aliases.

We may also count each term as an alias for itself given any set of equations. Then, although we have not yet stipulated all the conditions under which we will count terms as co-aliases, we have said enough to summarize the laws of reflexivity, symmetry, and transitivity—and more besides—by stating the following general principle:

Law for aliases: $\Gamma \vDash \tau = \upsilon$ if τ and υ are co-aliases given the set of equations in Γ (for any set Γ and terms τ and υ).

Like the law for a premise as a conclusion and a number of other principles we have used, the law for aliases gives sufficient but not necessary conditions for an entailment to hold, so it is stated with if rather than if and only if. To see why an equation can be a valid conclusion without its component terms being made co-aliases by the premises, note that, while an equation will be entailed by a set of equations only if it equates terms made co-aliases by that set, an equation can be entailed by a set of sentences without being entailed by the equations in the set. (For example, t = u is entailed by the premises $A \to t = u$ and A, and that set of premises contains no equations at all, only a conditional and an unanalyzed sentence.)

Linking a pair of terms by a chain of equations is not the only way a set might imply that they have the same extension. Recall the law of congruence for an *n*-place functor f

$$\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n \vDash \mathrm{f}\tau_1\ldots\tau_n = \mathrm{f}\upsilon_1\ldots\upsilon_n$$

This tells us that the terms $\mathrm{f}\tau_1\ldots\tau_n$ and $\mathrm{f}\upsilon_1\ldots\upsilon_n$ must have the same extension whenever this is true of their corresponding components (i.e., τ_1 and υ_1, τ_2 and υ_2, and so on). To incorporate this principle into the law for aliases, we will want to say that two applications of a given functor are made co-aliases whenever their corresponding components are made co-aliases, and we will want to allow this sort of connection between terms to figure as a link in a chain by which further terms are made co-aliases.

Putting all this together, we can give a fuller definition of the idea of co-aliases in the following way.

The *co-aliases given* a set Δ of equations include pairs of terms of all of the following kinds:
(i) a term paired with itself;
(ii) a pair of terms equated (in either order) by a member of Δ;
(iii) a pair of terms connected by a chain of terms linked as co-aliases given Δ;
(iv) a pair of applications of the same functor whose corresponding components are co-aliases given Δ.

Notice that the third and fourth classes are described in terms of the relation we are defining. A definition like this can be thought as a series of instructions for building the extension of the relation it defines. We first put in all the pairs covered by instructions (i) and (ii). Then we gradually add more pairs as we are directed to by instructions (iii) and (iv), replacing the phrase "co-aliases given Δ" by "pairs already in the extension." A pair of terms then count as co-aliases given Δ if and only if they are added at some stage in this process. And, since this process of building an extension for a two-place predicate can be described without using the term *co-alias*, we really have explained the meaning of that term.

In the simple examples we will usually consider, it will be easy to see which terms are co-aliases given a set of equations. But it may help in understanding the idea to think of the sort of "calculation" we might perform to apply the definition in a more complex example. When checking to see whether a pair of terms τ and υ are co-aliases given a set Δ of equations, let us collect all terms appearing as components in τ, υ, and Δ. Figure 6.3-2 shows these terms for a case where the set Δ consists of the equations a = b, fb = c, fb = fc, d = gca, and g(fa)b = e and we are checking to see whether the terms a and fd are co-aliases.

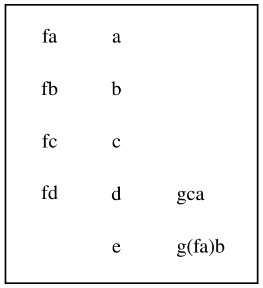


Fig. 6.3.2-2. A work space for finding co-aliases.

Notice that we include fa because it is a component of g(fa)b; however, there is no need to include fe or other more complex terms that could be formed from this vocabulary. (The arrangement of the terms is not significant; the one used here is designed simply to make later steps easier to depict.)

Now let us accumulate links between co-aliases. We will represent them as lines between terms. At the initial stage (which we will label 0), we put in links corresponding to equations in the set Δ. We can follow instruction (iii) after this and after each succeeding stage by considering terms to be co-aliases when they are linked either directly or by a chain, so there is no need to draw additional lines. At each of the stages from 1 on, we will consider all functors appearing among the terms and add any links we are directed to by instruction (iv); this will usually require new lines. We may need to do this several times over, but if we add no new links at any stage we can stop because there will be nothing to add thereafter. And, with a finite number of terms, this must happen at some point because there are only a finite number of links we might add.

Figure 6.3.2-3 shows such a process for the example of Figure 6.3.2-2, using labels on links to record the order in which they are entered. The new links at each stage are emphasized along with any older links that lead to the new entry. At stage 1, we check the applications of the functors f and g to see whether we can add any links by instruction (iv). Since a and b were already linked at stage 0, we add a link between fa and fb. We add no other links between the applications of f because d is not linked to a, b, or c. One pair of corresponding component terms from gca and g(fa)b (viz., a and b) were connected at stage 0 but the other pair (i.e., c and fa) were not, so the link between the two applications of g is entered only at stage 2 after c and fa have also been connected (by the link between fa and fb we enter at stage 1). Even at stage 2 the group including d is not linked to either the groups in which a, b, and c appear, so there are no further links between applications of f and the process is complete. The terms a and fd we were checking do not prove to be co-aliases at the end, but many other pairs of terms were shown to be co-aliases.

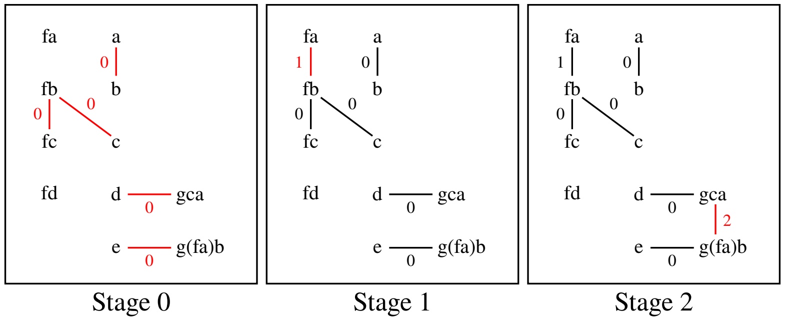


Fig. 6.3.2-3. Terms classified as co-aliases in a series of stages.

The links connect the terms in groups shown in Figure 6.3.2-4. The members of any group are co-aliases of one another but not of any other terms.

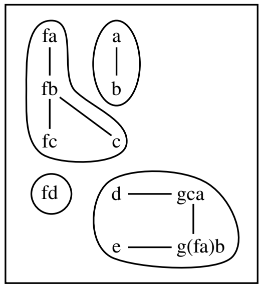


Fig. 6.3.2-4. Linked terms grouped in alias sets.

Some terms, like fd in the diagram, may be groups unto themselves; but, because they are co-aliases of themselves, we can still say that any pair made from such a group is a pair of co-aliases. If the terms had been written down more randomly, the links between them might have crossed and the groups of connected terms would no longer stand out; but they would still be there, and any diagram can be disentangled so that they appear. (This is a distinguishing feature of equivalence relations; any such relation divides a range of values into non-overlapping *equivalence classes*.) We will refer to each such group of connected terms as an *alias set*.

Now we are ready to justify our law of aliases, which claims that $\Gamma \vDash \tau = \upsilon$ whenever τ and υ are co-aliases given the equations in Γ. We can do this by showing how this law summarizes earlier ones. Each of the instructions (i)-(iv) for building connections between terms implements one or more of laws of entailment:

	instruction	*law(s)*
(i)	enter all terms appearing as components in τ, υ, and the set Δ of equations appearing in Γ	law of reflexivity (since entering the term establishes a link with itself)
(ii)	link each pair of terms equated (in either order) by a member of Δ	law for a premise as a conclusion and the law of symmetry (since a link amounts to an equation in both directions)
(iii)	count as linked any pair of terms connected by a chain of links	law of transitivity
(iv)	link any pair of applications of the same functor whose corresponding components are linked	law of congruence for functors

We combine several laws in (ii) and combine more by carrying out the instructions in a series of stages. This combination of laws can be justified by the law for lemmas because we can think of the process of adding links as a process of adding further equations as lemmas.

6.3.3. Derivations for identity

We are now in a position to state the derivation rules for identity that will be part of our basic system. We will have four rules for closing gaps. Each of them extends one of the rules QED and Nc, with each of those rules being extended in two ways. One sort of extension is based on the law for co-aliases alone and the other also rests on the law of congruence for predicates.

The first pair of extensions of QED and Nc—*Equated Co-aliases* (EC) and *Distinguished Co-aliases* (DC)—are shown in Figures 6.3.3-1 and 6.3.3-2.

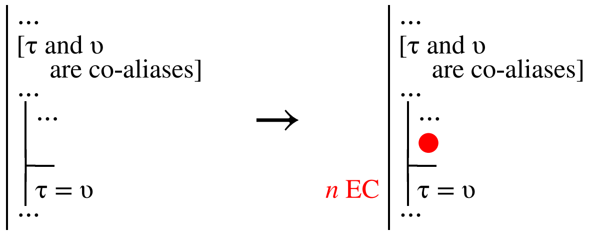


Fig. 6.3.3-1. Closing a gap whose goal is an equation between terms that are co-aliases with respect to the available resources.

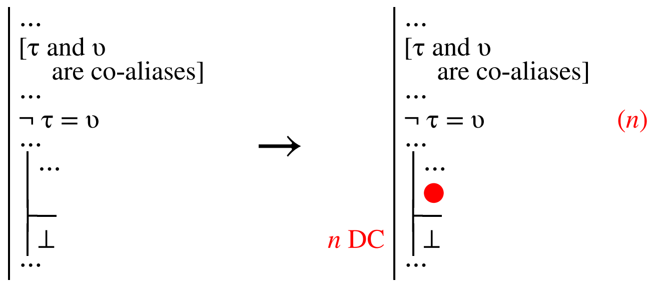


Fig. 6.3.3-2. Closing a gap of a *reductio* argument one of whose resources negates an equation between terms that are co-aliases with respect to the available resources.

The bracketed remark concerning τ and υ stipulates that there be enough equations among the available resources to make the terms τ and υ co-aliases. The law for aliases then tells us that the resources entail the equation $\tau = \upsilon$. So, if this equation is our goal, we may count the gap closed; and, if its denial is among our resources, we have the inconsistency required to close the gap of a *reductio* argument. An important special case of these rules is one where τ and υ are the same term. In this case, $\tau = \upsilon$ is $\tau = \tau$ (which is also $\upsilon = \upsilon$) and, since a term is a co-alias of itself with respect to any set—even with respect to a set in which it does not appear—any gap with a self-equation as its goal may be closed, as may the gap of a *reductio* argument with a negated self-equation among its resources. Notice that the general form of these rules differs from the special case for self-equations only by exchanging terms that are co-aliases.

Some abbreviated terminology will help in stating the second pair of rules for identity. Let us say that two series of terms $\tau_1 \ldots \tau_n$ and $\upsilon_1 \ldots \upsilon_n$ are *co-alias series* when they have the same length and their corresponding members are co-aliases—that is, when τ_i and υ_i are co-aliases for each i from 1 to n, where n is the length of the two series. Then the second pair of rules for identity are shown in Figures 6.3.3-3 and 6.3.3-4. These are *Quod Erat Demonstrandum Given Equations* (QED=) and *Non-contradiction Given Equations* (Nc=).

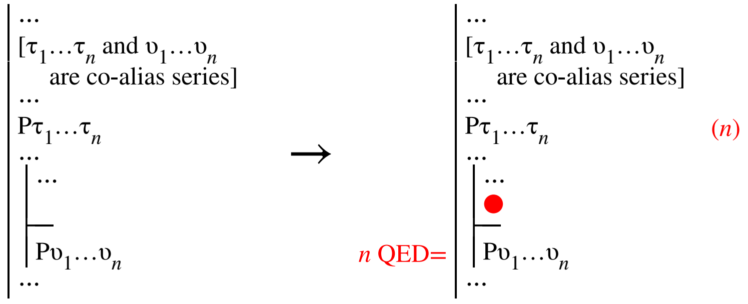


Fig. 6.3.3-3. Closing a gap one of whose resources differs from its goal only by terms that are co-aliases.

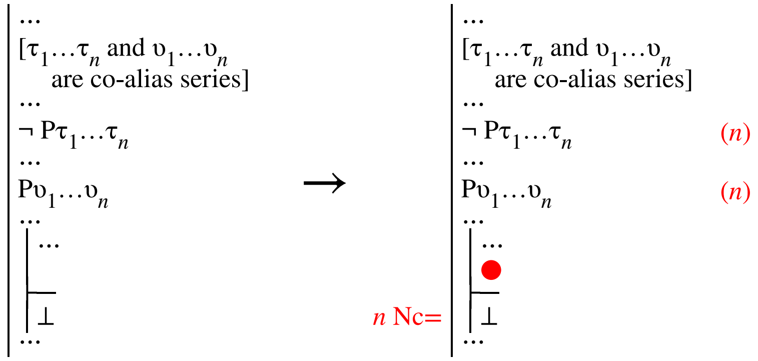


Fig. 6.3.3-4. Closing a gap of a *reductio* argument one of whose resources differs from the negation of another only by terms that are co-aliases.

Here a bracketed remark stipulates that the available resources contain enough equations to make corresponding component terms of $P\tau_1 \ldots \tau_n$ and $P\upsilon_1 \ldots \upsilon_n$ co-aliases and thus to entail identities between these terms. The law of congruence for P then tells us that $P\upsilon_1 \ldots \upsilon_n$ is entailed by available resources. If it is our goal, we may close the gap, and we may do so also if the gap is in a *reductio* argument and our resources contain its denial $\neg P\upsilon_1 \ldots \upsilon_n$.

The sentences $P\tau_1 \ldots \tau_n$ and $P\upsilon_1 \ldots \upsilon_n$ that figure in the last two rules have been described as applications of the same predicate whose corresponding component terms are co-aliases. A little thought will show that we can describe such expressions equally well as atomic sentences that differ only by components that are co-aliases. This makes the similarity of this rule to QED and Nc a little more apparent. Instead of saying that we can close a gap when our goal is among our resources and when one resource negates another (as we do in QED and Nc), we say here that we can close a gap if a resource differs from the goal or from the negation of another resource only by co-aliases. This way of describing these rules leads to the question whether we really need to limit them to predications. The answer is that we do not although that is the only case where we really need to use the rule.

Other rules can be extended in the way QED and Nc are extended in QED= and Nc=: if the illustration of the rule displays two occurrences of a sentence, these may be sentences that are different but that differ only by terms that are co-aliases given the available resources. (As was noted above, even our first two rules for identity could be seen as the result of extending in this way rules that say that we can close a gap whose goal is a self-equation and a *reductio* gap whose resources contain the denial of a self-equation.) When a rule is extended in this way, its label should be followed by the equals sign, as in the labels for QED= and Nc=. We will call the result an *extension* of the rule *for equations*; and, as with QED= and Nc=, the equals sign added to the name may be read "given equations."

Below are two derivations that illustrate these ideas and also show how we will keep track of co-aliases. The first derivation uses the rule of Figure 6.3.3-3 to close the gap after stage 2. The second uses an extended version of *modus ponens*; the resource Rc(fa) and the antecedent of Rcc → Gc differ only by terms (fa and c) that are co-aliases given the equations fa = b and b = c. (If the extended form of *modus ponens* was not used in the second derivation, we would need to set up an indirect proof to reach the goal Qc and exploit Rcc → Qc using the rule RC for exploiting conditionals in *reductio* arguments. Both gaps of the derivation would then close using the identity rules for closing gaps.)

```
         │ Fb ∧ a = b        1
         ├──
1 Ext    │ Fb                (3)
1 Ext    │ a = b             a—b, d
         │ │ a = d           a—b—d
         │ ├──
         │ │ ●
         │ ├──
3 QED=   │ │ Fd              2
         ├──
2 CP     │ a = d → Fd
```

```
         │ Rc(fa) ∧ fa = b   1
         │ Rcc → Qc          3
         ├──
1 Ext    │ Rc(fa)            (3)
1 Ext    │ fa = b            a, fa—b, c
         │ │ b = c           a, fa—b—c
         │ ├──
3 MPP=   │ │ Qc              (4)
         │ │ ●
         │ ├──
4 QED    │ │ Qc              2
         ├──
2 CP     │ b = c → Qc
```

At each stage when an equation is added to the resources, the resulting alias sets are indicated at the right. This is done by listing the members of each alias set with dashes between, separating the members of different alias sets by commas. This may be written to the right of the last equation added at a given stage. There is no need record the alias sets until the first stage when an equation appears as a resource since up to that point each term is in an alias set by itself.

The point of listing alias sets to the right of equations is to sum up the co-aliases at at each stage when they change. Although it is usually by adding equations that the alias sets will change, this is not the only possible way. When a term is added, either in a new resource or in a new goal, it must be accommodated in the co-alias sets. Although new terms will be introduced regularly in later chapters, they could be introduced now only if attachment rules or the rule LFR were used to introduce sentences that are not already components of sentences in the derivation. Since that is not a use of such rules that we have been considering, we will not consider examples, but a general guideline for listing alias sets can be stated that will include such cases: at the initial stage of a derivation if it has equations as resources and at any stage thereafter at which the alias sets of a gap have changed, list the alias sets at the right near the top of the gap. When several resources are added, the alias sets can be added after the last new resource that figures in the change. If no new resources are added (and the alias sets change only because of vocabulary added in a new goal), the alias sets may be listed at the right of the top of the scope line of the gap.

It is sometimes useful to be able to enter an equation between co-aliases as a further resource. Since this does not change the alias sets, it does bring a gap near an end and it is not automatically progressive. Therefore, we will count it as an attachment rule. We will call this rule *Co-alias equation* (CE):

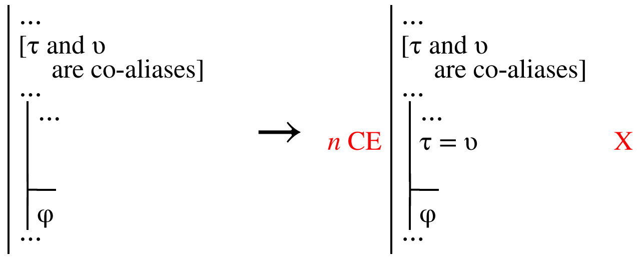


Fig. 6.3.3-5. At stage *n*, adding an equation between terms that are co-aliases with respect to the available resources.

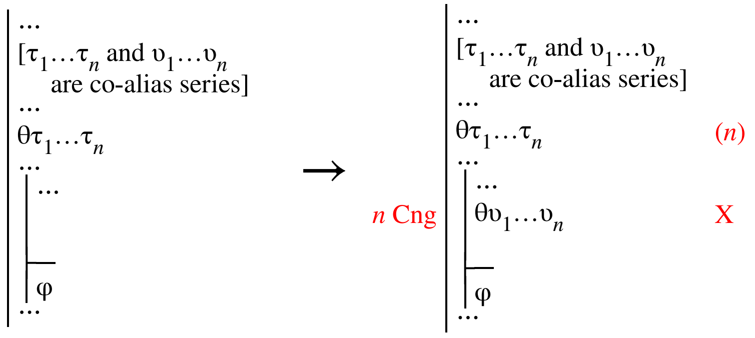
Equations are never exploited, so the X at the right does not mark the added equation as already exploited; instead it indicates that the equation leads to no change in the alias sets since its component terms are already co-aliases. Since any use of such an equation to close a gap is already covered by other rules, this rule will serve primarily to provide auxiliary resources for detachment rules (and available resources for use in other attachment rules). Here is a simple example.

```
         │ a = b
         │ b = c                      a—b—c, fa—fc, d
         │ (fa = fc ∧ d = d) → Ga     4
         ├──
1 CE     │ fa = fc                    X,(3)
2 CE     │ d = d                      X,(3)
3 Adj    │ fa = fc ∧ d = d            X,(4)
4 MPP    │ Ga                         (5)
         │ ●
         ├──
5 QED    │ Ga
```

In order to construct a derivation using only basic rules, we would need to resort to a *reductio* argument and the rule RC.

The rule CE is really only needed when no co-alias of the terms being equated appears as a term of an equation. The rule would not have been necessary in the example above if the conditional's antecedent had been something like a = c ∧ b = b because this sentence could be added by the extended attachment rule Adj= since it differs from a conjunction of the first two premises only by co-aliases. In general, if our resources contain any equation between co-aliases of the terms that we want to join in the new equation, we have an equation differing from the one we want only by co-aliases and we can use the extended form of whatever rule we might apply to the new equation.

Finally, we will add an attachment rule that allows a resource to be added when it differs from an available resource only by co-aliases. Such resources can be represented as $\theta\tau_1\ldots\tau_n$ and $\theta\upsilon_1\ldots\upsilon_n$ where $\tau_1\ldots\tau_n$ and $\upsilon_1\ldots\upsilon_n$ are co-alias series. That is, it is understood that the differences between the resources are limited to the displayed series of terms so the resources amount to predications to the two series $\tau_1\ldots\tau_n$ and $\upsilon_1\ldots\upsilon_n$ of an abstract θ that takes the form $[\ldots x_1 \ldots x_n \ldots]_{x_1\ldots x_n}$, where $x_1\ldots x_n$ is a series of distinct variables with the same length as $\tau_1\ldots\tau_n$ and $\upsilon_1\ldots\upsilon_n$. The name of the rule is *Congruence* (Cng).



…
[τ1…τn and υ1…υn are co-alias series]
…
θτ1…τn
…
…
φ
…
→
…
[τ1…τn and υ1…υn are co-alias series]
…
θτ1…τn
(n)
…
…
n Cng
θυ1…υn
X
φ
…

Fig. 6.3.3-6. At stage *n*, that differs from an available resource only by the occurrence of terms that are co-aliases.

The resource that is added by this rule is marked as exploited because any exploitation of the earlier resource will be enough to take account of it. The rule Cng offers the following alternative to an earlier derivation.

```
        │Rc(fa) ∧ fa = b   1
        │Rcc → Qc          4
        ├─
1 Ext   │Rc(fa)            (3)
1 Ext   │fa = b            a, fa—b, c
        │
        │ │b = c           a, fa—b—c
        │ │
3 Cng   │ │Rcc             X,(4)
4 MPP   │ │Qc              (5)
        │ │●
        │ ├─
5 QED   │ │Qc              2
        ├─
2 CP    │b = c → Qc
```

Notice that the ordinary form of MPP is used here rather than the extended form MPP= used earlier, and Cng can always be avoided by using the extended forms of other rules. The point of using Cng is only to add a step to a derivation that may make it easier to follow.

6.3.s. Summary

1 The logical properties of identity have two sources, the extension stipulated for = and the requirement that all predicates and functors be extensional. We will approach these properties by speaking of the terms equated by a true equation as co-aliases. Some thought about this idea shows us that identity obeys laws of reflexivity, symmetry, and transitivity, so it is an equivalence relation. Identity is distinguished as holding in the fewest cases of any equivalence relation; it implies sameness in respect to all predicates and functors. That is, identity is a congruence for each predicate and functor. To say that identity is a congruence for a predicate or functor is to say that it is an extensional operation. A predicate that did not satisfy this requirement would be an intensional property (as distinct from a property in intension, which is the meaning of an ordinary extensional predicate) and the things of which it was true or false would be intensional entities. Whether these ideas are needed to account for aspects of deductive reasoning (or are even coherent) has been a matter of controversy, but we will consider only extensional operations.

2 A different way of organizing the laws for identity is useful in stating derivation rules. We say that terms are co-aliases given a set Δ of equations if an equation between the terms follows from Δ. A set Δ of equations serves to divide a collection of terms into alias sets, groups of terms whose members are mutual co-aliases; these are examples of the equivalence classes associated with any equivalence relation. The alias sets determined by a given set of equations can be found by a process of making links between terms, following rules that implement the laws for identity. As a result, identity obeys a law for aliases that says that an equation $\tau = \upsilon$ is entailed by a set of premises if the terms τ and υ are co-aliases given the equations among those premises.

3 The law for aliases and the law of congruence for predicates provide us with the basic derivation rules for =, each of which is a rule for closing gaps. The rules employ the idea of terms being co-aliases given the equations among the resources of a gap. One rule, Equated Co-aliases (EC), says a gap may be closed if its goal is an equation between co-aliases, and another, Distinguished Co-aliases (DC), says a *reductio* gap may be closed if its resources include a denial of such an equation. A second pair concern predications of the same predicate to series of terms whose corresponding members are co-aliases. One of these, QED Given Equations (QED=) says

that a gap may be closed if its goal is a predication that differs from another predication among the resources only by co-aliases and another, Nc Given Equations (Nc=) says that a *reductio* gap may be closed if one of its resources differs from what another denies only by co-aliases. The statements of these rules use the idea of co-alias series of terms, two series of the same length whose corresponding terms are co-aliases. The idea behind this second pair of rules can be carried further and we may extend any rule by counting as identical, for the purposes of applying the rule, any sentences that differ only by terms that are co-aliases. There are two attachment rules for identity that may be convenient. One, Co-alias Equation (CE), allows us to add to the resources any equation between co-aliases and the other, Congruence (Cng), allows us to add a predication that differs only by co-aliases from one already among the available resources.

Glen Helman 03 Aug 2010

6.3.x. Exercise questions

Use the system of derivations to establish each of the following:

1. Fa → Ga, Fa, a = b ⊨ Gb

2. Fa → Ga, Fb, a = b ⊨ Ga

3. Fa ∧ a = gb ⊨ ¬ F(gc) → ¬ b = c

4. Fa → G(fa), G(fb) → Hb, a = b ⊨ Fb → Ha

5. fa = b, fc = d ⊨ (a = c ∨ b = d) → fa = d

6. The vice president is Joe Biden
Barack Obama is the president
The vice president is not from Illinois
____
If Barack Obama is from Illinois, then Joe Biden is not the president

For more exercises, use the exercise machine.

Glen Helman 03 Aug 2010

6.3.xa. Exercise answers

Some of the derivations below are given twice, once using only the basic identity rules EC, DC, QED=, and Nc= and a second time using MPP= and similar extensions for equations of other rules (see 6.3.3); either approach is entirely acceptable.

1.

```
          │Fa → Ga   1
          │Fa        (1)
          │a = b     a—b
          ├─
1 MPP     │Ga        (2)
          │●
          ├─
2 QED=    │Gb
```

2.

```
          │Fa → Ga   2
          │Fb        (3)
          │a = b     a—b
          ├─
          ││¬ Ga     (2)
          │├─
2 MTT     ││¬ Fa     (3)
          ││●
          │├─
3 Nc=     ││⊥        1
          │
1 IP      │Ga
```

```
          │Fa → Ga   1
          │Fb        (1)
          │a = b     a—b
          ├─
1 MPP=    │Ga        (2)
          │●
          ├─
2 QED     │Ga
```

3.

```
          │Fa ∧ a = gb            1
          ├─
1 Ext     │Fa                     (4)
1 Ext     │a = gb                 a—gb, b, c, gc
          │
          ││¬ F(gc)               (4)
          │├─
          │││b = c                a—gb—gc, b—c
          ││├─
          │││●
          ││├─
4 Nc=     │││⊥                    3
          │├─
3 RAA     ││¬ b = c               2
          ├─
2 CP      │¬ F(gc) → ¬ b = c
```

4.

```
          │Fa → G(fa)    3
          │G(fb) → Hb    5
          │a = b         a—b, fa—fb
          ├─
          ││Fb          (4)
          ││
          │││¬ Ha       (7)
          ││├─
          ││││●
          │││├─
4 QED=    ││││Fa        3
          │││
          ││││G(fa)     6
          ││││├─
          │││││●
          ││││├─
6 QED=    │││││G(fb)    5
          ││││
          │││││Hb       (7)
          │││││●
          ││││├─
7 Nc=     │││││⊥        5
          │││├─
5 RC      ││││⊥         3
          ││├─
3 RC      │││⊥          2
          │├─
2 IP      ││Ha          1
          ├─
1 CP      │Fb → Ha
```

```
          │Fa → G(fa)    2
          │G(fb) → Hb    3
          │a = b         a—b, fa—fb
          ├─
          ││Fb          (2)
          │├─
2 MPP=    ││G(fa)       (3)
3 MPP=    ││Hb          (4)
          ││●
          │├─
4 QED=    ││Ha          1
          ├─
1 CP      │Fb → Ha
```

5.

```
          │fa = b
          │fc = d                          a, b—fa, c, d—fc
          ├─
          ││a = c ∨ b = d                  2
          │├─
          │││a = c                         a—c, b—fa—fc—d
          ││├─
          │││●
          ││├─
3 EC      │││fa = d                        2
          ││
          │││b = d                         a, fa—b—d—fc, c
          ││├─
          │││●
          ││├─
4 EC      │││fa = d                        2
          │├─
2 CP      ││fa = d                         1
          ├─
1 CP      │(a = c ∨ b = d) → fa = d
```

6.

```
         │v = b
         │o = p             o—p, b—v
         │¬ Fvi             (3)
         ├─
         ││Foi             (3)
         │├─
         │││b = p          o—p—b—v
         ││├─
         │││●
         ││├─
3 Nc=    │││⊥              2
         │├─
2 RAA    ││¬ b = p         1
         ├─
1 CP     │Foi → ¬ b = p
```

F: [_ is from _]; v: the vice president; b: Barack Obama; c: Joe Biden; p: the president; t: Illinois

6.4. Describing models

6.4.0. Overview

The grammatical variety we have been considering brings with it a greater variety of semantic values; in particular, the counterexamples to arguments that are not formally valid are now more than simple assignments of truth values.

6.4.1. Extensions and ranges

While the extensions of individual terms are (like the extensions of sentences) single values, the extensions of operations are (like the extensions of connectives) functions.

6.4.2. Building structures

The extensions of predicates (functions from reference values to truth values) can be fully specified by telling the input for which they give output T, and these cases can be presented in diagrams to which the extensions of other items of non-logical vocabulary can be added.

6.4.3. Structures as counterexamples

Although extensional interpretations are now different, it is still true that a dead-end open gap specifies the sort of interpretation that divides the gap.

6.4.1. Extensions and ranges

In this section, we will look at ways of describing the semantic values of the new sorts of expression we have been considering and ways of using these values to present counterexamples to derivations that fail. First, let us collect and sharpen what we know about the semantic values of the several kinds of expression we are considering. Table 6.4.1-1 gives a basic summary that you may compare with the tables of grammatical categories given in 6.1.1 and 6.1.7.

Expression	*Extension*		
sentence	truth value		
term	reference value		
		input	*output*
connective	truth function	truth value(s)	truth value
predicate	property or relation	reference value(s)	truth value
functor	reference function	reference value(s)	reference value

Table 6.4.1-1. The extensions of 5 kinds of expression.

In each case the *intension* of an expression is a specification of its extension in each possible world. For example, the intension of an individual term is specifies its reference value in each possible world; this is the sort of intensional entity that was mentioned in 6.3.1. In particular, while Barack Obama and the U. S. president have the same extension in the actual world, they have different intensions because their extensions differ in other possible worlds.

Since the extensions of the incomplete expressions are functions, they exhibit generality: each such extension determines an output value for each input value from some range of such values. In the case of connectives, the input values are fixed as the two truth values **T** and **F**, and the range of generality of truth functions is thus quite limited. We do not fix the range of reference values, but this range must be known before we know what functions are available as extensions of predicates and functors. We will refer to a specification of the reference values as a *referential range* or often simply as a *range*, and we will use the symbol **R** for it. (The word domain is often used for this idea, but we will use that word for another concept.) The referential range can be any set that is not empty.

Our logical constants have fixed extensions that we stipulate once and for all. In the case of connectives these are given by their tables. The identity predicate = has an extension that is settled once the referential range is settled: this predicate is true of any pair of reference values whose members are the same but false of any pair of different values. Further basic expressions —unanalyzed sentences, unanalyzed terms other than variables, unanalyzed predicates, and unanalyzed functors—form our *non-logical vocabulary*, and their extensions are not fixed.

As in truth-functional logic, items of non-logical vocabulary may be assigned extensions by *extensional interpretations* or assigned extensions for all possible worlds by *intensional interpretations*. The extensions assigned to predicates and functors by a given interpretation must have a generality that extends to the same range **R**, so we will speak of an extensional interpretation as being an *interpretation on* a range **R**. The basic semantic information needed for the logical forms we are now considering is then a range **R** and an extensional interpretation (on that range) of certain items of non-logical vocabulary; we will refer to this information as a *structure for* any expressions that can be formed from this non-logical vocabulary and our logical vocabulary.

It will be convenient to assume that every reference value in the range of a structure comes with a label. We will refer to this label as the *ID* of the value. The assumption that all reference values have IDs is actually quite a heavy one. The limitations of decimal notation in capturing irrational numbers like π and the square root of 2 are essential; no system of finite expressions could name all real numbers. So if a range includes all real numbers, its members could not all be labeled by expressions in any ordinary sense. One way around this is to think of IDs as mathematical abstractions—for example, as ID numbers that are not merely numerical expressions but genuine numbers. In this way, the real numbers might be used as their own IDs. But, while these are important theoretical issues that have had considerable impact on the development of logic, they will not affect us practically. Our chief interest will be in structures indicated by the dead end gaps of derivations; and these will all have finite (and usually very small) ranges, so there will be no problem in using numerals (even single-digit numerals) as their IDs.

6.4.2. Building structures

Once a referential range **R** is specified, the extensions of the various sorts of non-logical vocabulary can be specified in ways that extend the approach used in truth-functional logic. Individual terms are merely assigned reference values from **R** in the way sentences were assigned truth values. The extensions of predicates and functors are functions and, as we have already seen, these can be indicated by tables analogous to truth tables. Below is an example of an extensional interpretation of the following non-logical vocabulary:

sentences:	A, B
individual terms:	a, b, c, d, e
predicates:	F (*1-place*), G (*1-place*), R (*2-place*)
functors:	f (*2-place*)

We choose (arbitrarily) a referential range with five values whose IDs run from 0 to 4 and assign (again arbitrarily) extensions of the appropriate sorts to the items of non-logical vocabulary.

R: 0, 1, 2, 3, 4

A	B
T	F

a	b	c	d	e
3	4	0	2	4

τ	Fτ
0	F
1	T
2	T
3	F
4	T

τ	Gτ
0	T
1	F
2	F
3	T
4	T

R	0	1	2	3	4
0	F	F	F	F	F
1	T	F	T	F	T
2	F	T	F	F	T
3	F	T	F	T	F
4	F	F	F	F	T

f	0	1	2	3	4
0	1	2	3	1	4
1	2	4	0	1	3
2	3	1	2	0	1
3	4	1	0	3	0
4	4	3	1	2	4

The truth-table row giving the values of A and B is dwarfed by the other information in the structure, but the whole of the structure has the same significance for our present analysis of logical form as did the left side of a single row of a truth table in truth-functional logic.

Since we build in no assumptions about the size of the range **R**, we can consider structures that are quite small, and it is possible to represent small structures in pictorial diagrams. As in Figure 6.4.2-1, let us depict a range by a rectangle, with the values of the range shown as circles that enclose numbers, which will serve as the IDs of the reference values in the range.

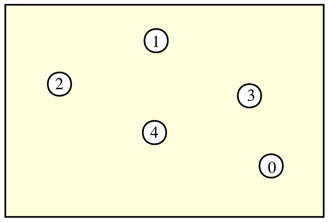


Fig. 6.4.2-1. A range of reference values labeled by their IDs.

The extension of a simple term will be one of these reference values, and we can show this by writing the term next to that value. Figure 6.4.2-2 shows the extensions assigned above to the terms a, b, c, d, and e. The terms b and e are written next to the same value because both were assigned that value as their extension.

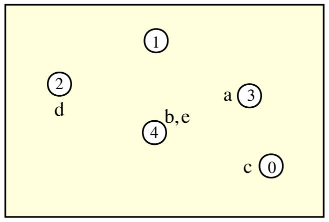


Fig. 6.4.2-2. A range with the extensions of five terms shown; two have the same extension.

The extension of a one-place predicate is a function that yields a truth value as output when it is applied to a reference value as input. We will say that it is true or false *of* a reference value depending on its output for that value as input. We can represent this sort of extension by writing the predicate next to the values it is true of; we will then know that it is false of any other reference values. Figure 6.4.2-3A does this for the predicates F and G, again using the extensions originally given in tables.

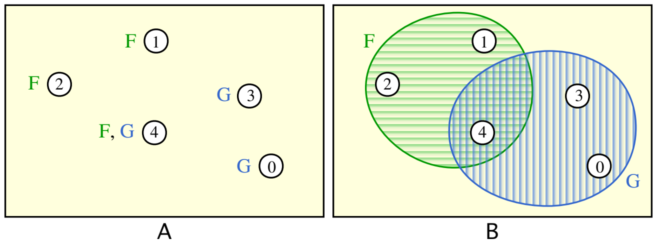


Fig. 6.4.2-3. A range with the extensions of two one-place predicates indicated by labeling values (A) and enclosing sets of values (B).

This can make the diagram rather cluttered, but we can clean things up a little by drawing a line around all the values a predicate is true of and labeling the line rather than the values. This is done in Figure 6.4.2-3B. The second sort of diagram corresponds fairly directly to what was the original concept of an extension—the class of things a predicate is true of—and we can say that the values a predicate is true of are *in* its extension.

Predicates with more than one place are not true or false of single values but of pairs, triples, or longer series of values. For example, [_ is the father of _] is true not of James Mill or of John Stuart Mill (his son) taken individually but instead of the two taken together and in that order. Such an *ordered pair* of values can be represented in our diagrams by an arrow from its first to its second member. (We could represent longer series of values by adding legs between the head and tail of the arrow.) The extension of a 2-place predicate can be thought of as the collection of pairs it is true of. Similarly, the extension of a 3-place predicate will be a collection of triples, and the extension of a predicate with some number *n* of places will be a collection of *n-tuples* (i.e., of series with length *n*).

There are a number of ways a collection of *n*-tuples can be depicted. We might draw the arrows that represent its members and label each one as we initially labeled the values in the extension of a one-place predicate. This would make for some more clutter, but it would be hard to avoid that by drawing a line around a group of arrows. We might write the predicate once and draw a line from it to each of the arrows in its extension, or we might draw different styles of arrows for different predicates, labeling the style of arrows in a legend like that of a road map. Each of these three approaches to labeling the extensions of many-place predicates has its value but we will most often use legends. Figure 6.4.2-4 shows this latter style of diagram for the extension that was assigned to the 2-place predicate R. Notice that the predicate is true of the values 1 and 2 taken in either direction and that it is also true of each of 3 and 4 paired with itself (and look back to see how that information appeared in the table).

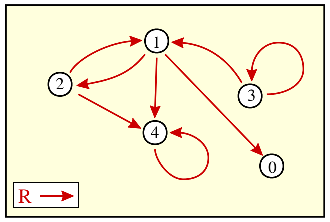


Fig. 6.4.2-4. A range with the extension of a 2-place predicate indicated.

Figure 6.4.2-5A combines the extensions of the three predicates. As was noted in 6.2.2, unanalyzed sentences can be thought of as zero-place predicates. That means that they do not express properties that may or may not be true of objects or relations that may or may not hold between objects. Instead sentence express state of affairs that are simply true or false. One way to indicate that in a diagram is to simply include a true sentence within the rectangle, understanding any unanalyzed sentence that does not appear there to be assigned the value **F**. That addition to the diagram is shown in Figure 6.4.2-5B.

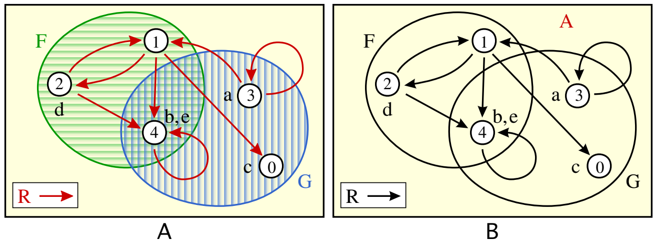


Fig. 6.4.2-5. A range with the extensions of predicates (A) and predicates together with the indication that a sentence is true (B).

All that is left are the extensions of functors. We could use arrows here, too, because a reference function establishes a relation between its input and output values. For example, the squaring function relates 1 to itself, 2 to 4, 3 to 9, and so on. However, this would make for a lot of arrows. A one-place function relates each value to some other value (perhaps the nil value), so each value would be at the tail of an arrow; and things get much worse with functions of two or more places. We can get a somewhat more compact notation by adapting the way we indicate the extensions of individual terms. Next to each output value of a functor, we can write the functor with its places filled by IDs of the input values for which it yields that output (for example, writing f01 next to the value 2 to say that 2 is the output of f for inputs 0 and 1). Since the extension of a functor may yield the same output for different input values, we may need to write the functor next to an ID several times, each time filling its places with the IDs of different input values. This is manageable for 1-place functors because, for each such functor, we will need only as many labels as there are possible input values—i.e., one for each member of the range. But for a function with two places, the number of labels is the square of the number of reference values and this number mounts rapidly. In the example we are considering, we would need to write the 2-place functor f 25 times to indicate all 25 entries of the table shown earlier. Adding to these only the individual terms leads to the rather cluttered diagram shown in Figure 6.4.2-6.

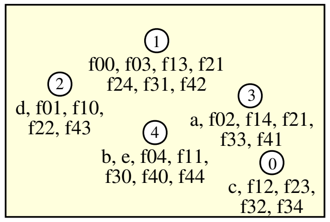
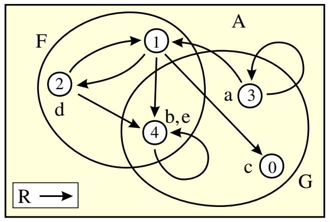


Fig. 6.4.2-6. A range showing the extensions of a 2-place functor and several individual terms.

Most of the structures we consider will be quite small, and this approach will be more feasible with them. Still, it is always possible to supplement a diagram with one or more tables, and that is the easiest approach for the example we have been considering. The full interpretation is given in this way in Figure 6.4.2-7.



f	0	1	2	3	4
0	1	2	3	1	4
1	2	4	0	1	3
2	3	1	2	0	1
3	4	1	0	3	0
4	4	3	1	2	4

Fig. 6.4.2-7. A structure for a variety of non-logical vocabulary.

Now let us do something with this structure. Interpretations are assigned in order to settle the truth values of sentences formed using the vocabulary that is interpreted. This is analogous to calculating a truth value of a truth-functional compound given an assignment of truth values to its ultimate components. Below is the calculation of the truth value given to a sentence by the structure we have been considering. It is followed by an explanation of the initial steps in the process; this explanation refers to the way the interpretation is presented in the diagram and table in Figure 6.4.2-7.

```
(Ga ∧ Rde) → ((F(fab) ∨ ¬ B) ∧ b = e)
 T3 T T24   Ⓣ    F 034  T T F   T 4 T 4
```

Ga: a has 3 as its value and this value is in the area representing the extension of G, so Ga gets **T**

Rde: d and e have 2 and 4 as their extensions and the arrow for this pair is in the extension assigned to R, so Rde gets **T**

F(fab): f yields the value 0 when given the extensions of a and b (the values 3 and 4) as input (as can be seen from the end of the next-to-last row of the table for F), and 0 is not in the area marked as the extension of F; thus fab gets 0 and F(fab) gets **F**

B: B gets the value **F** since, unlike A, it does not appear within the rectangle

b = e: b and e both have 4 as their extension, so b = e gets **T**

The extensions of complete unanalyzed expressions have been written under these expressions, and the values of compounds are written under signs for the operations that form them. As in truth-functional logic, the order of calculation is determined by parentheses. Notice that capital letters always have truth values under them and lower case letters always have reference values under them.

6.4.3. Structures as counterexamples

Since structures provide the information that is now needed to determine truth values for sentences, we will present counterexamples to derivations that fail by describing structures. An example of a failed derivation is shown below.

```
         │P(fa)b → Qa(fd)          3
         │Qbd → Fb                 5
         │b = d                    a, b—d, fa, fd
         ├─
         ││P(fd)d ∧ a = d          2
         │├─
2 Ext    ││P(fd)d                  (3)
2 Ext    ││a = d                   a—b—d, fa—fd
3 MPP=   ││Qa(fd)
         ││
         │││¬ Fd                   (5)
         ││├─
5 MTT=   │││¬ Qbd
         │││○                      b=d,P(fd)d,a=d,Qa(fd),¬ Fd,¬ Qbd ⊭ ⊥
         │││
         │││⊥                      4
         │├─
4 IP     ││Fd                      1
         ├─
1 CP     │(P(fb)d ∧ a = d) → Fd
```

Stage 3 of the development uses the extended version of *modus ponens*. At this point, we have two alias sets, one consisting of a, b, and d and the other consisting of fa and fd. We do not have the antecedent of the conditional P(fa)b → Qa(fb) among our resources but rather a sentence, P(fd)d, that, although differing from it in two places, differs only by terms that are co-aliases for fa and b. Stage 5 uses a similarly extended *modus tollens*. The remaining open gap cannot be closed because Qa(fd) and ¬ Qbd, the two resources that might be part of a contradiction, differ in their second place by terms (fd and d) that have not been made co-aliases.

The active resources of the dead-end gap form the consistent set:

b = d, P(fd)d, a = d, Qa(fd), ¬ Fd, ¬ Qbd

To describe a structure making the members of this set true, we must choose a range of reference and assign an extension to each of the items of non-logical vocabulary. The choice of the referential range and the assignment of extensions to both individual terms and functors is determined by the alias sets. We choose one reference value for each alias set and assign extensions so that the terms in the set have that as their reference value.

For this consistent set, we will have two alias sets, one containing a, b, and d and the other containing fa and fd, so we take the range to consist of two values, one corresponding to each alias set. We do this by numbering the alias sets and taking these numbers to be the IDs of the values in the range.

Next we must assign values to non-logical vocabulary appearing in the terms in such a way that each term has the reference value corresponding to the number of its alias set. In the case of an unanalyzed term we simply assign the value of its alias set. In the case of a compound term, we place the following constraint on the interpretation of its main functor (the one used last in forming it): the output must be the value associated with the alias set of the compound when the input consists of the reference values associated with the alias sets of the component terms. In the example we are looking at, the two compound terms place the same constraint since they are co-aliases and have components which are co-aliases. The table below shows the association of ID numbers with alias sets and the constraints on the structure that follow from this association:

term	*ID*	*constraint*
a	1	a: 1
b		b: 1
d		d: 1
fa	2	f1: 2
fd		f1: 2

To indicate constraints, we use a variant of the notation used to indicate the extensions of functors in the diagrammatic presentation of structures. Here “f1: 2” says that interpretation of f must yield output with ID 2 for input with ID 1.

We also have three non-logical predicates to consider, the 2-place predicates P and Q and the 1-place predicate F. Each sentence in the consistent set that affirms or denies one of these of a series of terms provides a constraint on the interpretation of that predicate—as is shown in the following table.

resource	*constraint*
P(fd)d	P21: **T**
Qa(fd)	Q12: **T**
¬ Qbd	Q11: **F**
¬ Fd	F1: **F**

The sentence P(fd)d tells us that P is true of values 2 and 1 (in that order) since these are the values of fd and d, respectively; but no other sentence says anything about the extension of P. There are sentences that require that the predicate Q be true of the pair 1 and 2 and false of the pair 1 and 1, but nothing is said about other cases. The last sentence requires that F be false of 1 but requires nothing beyond this.

The tables below incorporate this information about extensions. The values in grey are not required to make the members of the consistent set true and may be assigned arbitrarily. In the case of predicates, the value **F** has been assigned in such cases to make the extension as small as possible.

R: 1, 2

a	b	d
1	1	1

τ	fτ
1	2
2	1

τ	Fτ
1	F
2	F

P	1	2
1	F	F
2	T	F

Q	1	2
1	F	T
2	F	F

The upshot of these tables is depicted in Figure 6.3.4-1.

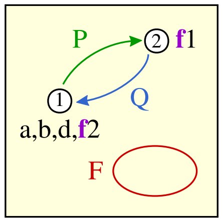


Fig. 6.4.3-1. A structure dividing the open gap of the derivation above.

Since the predicates P and Q are each true of only one pair, they are used to label arrows directly. The emptiness of F's extension is shown by using F to label a circle that encloses nothing. This structure is small enough that the extension of the functor f is also represented in the diagram.

Much of the work here comes in assigning interpretations to individual terms and functors on the basis of a collection of alias sets. Let us look at another example of that. The example we worked out in 6.3.2 would arise if we were to check the entailment

$$a = b,\ fb = c,\ fb = fc,\ d = gca,\ g(fa)b = e \vDash a = fd$$

The derivation for this is not very interesting. A single use of IP would leave us with a dead-end open gap which fails to close because

$$a = b,\ fb = c,\ fb = fc,\ d = gca,\ g(fa)b = e,\ \neg\, a = fd \nvDash \bot$$

The alias sets we found in 6.3.2 are shown below along with the corresponding constraints on the interpretation of individual terms and functors:

term	*ID*	*constraint*
a	1	a: 1
b		b: 1
c	2	c: 2
fa		f1: 2
fb		f1: 2
fc		f2: 2
fd	3	f4: 3
d	4	d: 4
e		e: 4
gca		g21: 4
g(fa)b		g21: 4

As in the example above, an unanalyzed term is simply assigned the number of its alias set. For a compound term, we require that the number of the alias set be the output value corresponding to input(s) that are the numbers of the alias sets of its immediate components. For example, the term fa appears in set 2, so we want the table for f to lead us to calculate 2 as the reference value of fa. The input for the calculation will be the reference value of the term a; but a appears in set 1, so we want the table for f to yield output 2 for input 1. We derive exactly the same information from the appearance of the term fb; the output is the same because it appears in the same alias set as fa, and the input is the same because the term b appears in the same alias set as the term a. On the other hand, the appearance of fc in alias set 2, tells us that the table for f should assign output 2 also for input 2 since 2 is the alias set of the term c. We respond to the remaining terms in a similar way, the only difference being the need to note pairs of input values in the case of the 2-place functor g.

When we put constraints in tables assigning extensions to the individual terms a, b, c, d, and e the functors f and g, we get the following:

R: 1, 2, 3, 4

a	b	c	d	e
1	1	2	4	4

τ	fτ
1	2
2	2
3	
4	3

g	1	2	3	4
1				
2	4			
3				
4				

Many entries are left unfilled because they did not correspond to any terms in our alias sets. But, by the same token, we will never use these entries to calculate the values of terms appearing in the open gap, so they can be filled in arbitrarily. The value 1 is used in the tables below but any other would do; it is the other values that are significant.

R: 1, 2, 3, 4

a	b	c	d	e
1	1	2	4	4

τ	fτ
1	2
2	2
3	1
4	3

g	1	2	3	4
1	1	1	1	1
2	4	1	1	1
3	1	1	1	1
4	1	1	1	1

Recall that, in a couple of cases, we have had a single input-output pair dictated by two different terms. This raises the question whether the procedure we are using could ever lead to impose incompatible requirements? That is, could we end up trying to associate two different output values of a functor with the same series of input values and thus to fill in one entry in two different ways? For this to happen, there would have to be terms $f\tau_1 \ldots \tau_n$ and $f\upsilon_1 \ldots \upsilon_n$ with a common functor f that fell into different alias sets (if we were to have two output values), and the corresponding components of these compounds (τ_i and υ_i for *i* from 1 to *n*) would have to fall in the same alias sets (if we were to have the same input values in the two cases). But the way we have set up alias sets insures that this cannot happen. Instruction (iv) for drawing links would have told us to put the two compounds in the same alias set once their corresponding components were connected. And, indeed, in the two cases where we have duplicate requirements, the compounds appear in the same alias set precisely because we followed this instruction when forming the alias sets of this example. (Although terms whose corresponding components are co-aliases are bound to appear in the same alias set, they might do so for other reasons, too; for example, we might have both a = b and fa = fb as resources of a gap we are trying to divide.)

We have now done enough to settle the truth values of all equations that appear affirmed or negated among the premises we are trying to make true. Do these values come out as we would like? That is, do the affirmed equations come out true and the negated ones false? Well, since the extensions given to all terms, simple or compound, will correspond to their alias sets, we know that any equation $\tau = \upsilon$ that is affirmed among the premises will be true. For such an equation will have led us to put the terms τ and υ into the same alias set, and each term will be assigned the value corresponding to this set as its extension. And, since they have the same extension, the equation between them will be true. How about the denial of an equation, a resource of the form $\neg\, \tau = \upsilon$? Since the gap cannot be closed, we know that τ and υ are members of different alias sets. And since the extensions given to these terms correspond to their alias sets, they will have different reference values and the equation $\tau = \upsilon$ will be false, making the resource $\neg\, \tau = \upsilon$ true—as is the case with ¬ a = fd in the example above.

We have been focusing on functors and equations since that is all that matters for the example, but similar considerations apply to non-logical predicates and predications of them. In the case of such predicates, it is our rules for closing gaps insure that we can assign interpretations consistently. If the gap cannot be closed we know that it does not contain both $P\tau_1 \ldots \tau_n$ and any sentence $\neg\, P\upsilon_1 \ldots \upsilon_n$ where the corresponding terms are co-aliases. And this means it never contains both an affirmation and a denial of P of any series of terms whose corresponding members are in the same alias sets. This means that we will never be led to require the extension of P to yield two different outputs for the same input. And the requirements we place on the extensions of non-logical predicates are designed to insure directly the truth of sentences affirming or denying the predication of such a predicate, so it is enough to know that our requirements are consistent to be sure that they will have the desired result.

The procedure we have been following enables us to find a structure dividing any dead-end open gap, and the safety of our rules tells us that the same structure will divide the initial premises and conclusion of the derivation. Now the existence of a structure dividing premises and conclusion is the test of formal validity of an argument. That is, if there is a structure that divides an argument's premises from its conclusion, then there is an intensional interpretation of it producing an actual English argument and a possible world that will divide the premises and conclusion of that argument. This was easy to see in truth-functional logic, but more needs to be said in the case of the more complex interpretations we are now considering.

We cannot, as in 2.3.1, simply choose the actual world as the possible world that divides premises from conclusion because a structure, such as the one in Figure 6.4.3-1, may have only a limited number of reference values, while the actual world has many things in it (infinitely many if numbers are counted). The easiest approach in the present setting (but one that will no longer work in the next chapter) is to note that our calculations of extensions for the terms we are interested in remain the same in the presence of further reference values. When we chose a referential range, we could have added reference values that did not correspond to alias sets. Such values would not have played a role in the constraints on the interpretation of non-logical vocabulary or in the calculations of the values of components of the premises and conclusion of the argument we are interested in. So they would have neither contributed to nor interfered with the task of dividing the premises from the conclusion. The possibility of adding such further reference values means that we can regard a structure like that of Figure 6.4.3-1 as a depiction of the way things stand for certain reference values *among others*. Given this understanding of a structure, it is not too hard to concoct intensional interpretations of the non-logical vocabulary that have the right extensions in the actual world. We might, for example, choose language describing an illustration of the structure. To capture the structure of 6.4.3.1, the interpretation of the term a could be the point labeled a and the interpretation of P could be [a P-arrow runs from _ to _]. If we use this sort of interpretation, drawing the structure is a way of making the actual world divide the argument's premises from its conclusion.

Glen Helman 03 Aug 2010

6.4.s. Summary

1 Logical forms (without free variables) may be given semantic values by assigning values to the non-logical vocabulary they contain; that is, they can be given extensions (or intensions) by an extensional (or intensional) *interpretation* of this vocabulary. The extensions of predicates and functors are functions that take as input reference values from a referential range **R** that must be specified along with an extensional interpretation; the range and the interpretations of non-logical vocabulary together constitute a structure for any expressions formed using only the non-logical vocabulary that is interpreted in the structure. We assume each value of the range is labeled by an ID.

2 The extensions of non-logical vocabulary can be represented using tables. In a more graphic approach, a referential range may be depicted by points in a plane labeled by their IDs, and further labeling and other devices can depict extensions of non-logical vocabulary on this range. For example, one-place predicates may label the points they are true of either individually or by labeling a line enclosing them. This set of points is one way of representing the extension of the predicate. If a predicate has more than one place, its extension must be a set of ordered pairs, triples, or other *n-tuples*; these may be represented by arrows (perhaps with legs) that indicate the order of values in the *n*-tuple. We may calculate the extensions that structures give to expressions by using a table analogous to a truth table, with all the information in a structure providing the basis for the calculation of a single row.

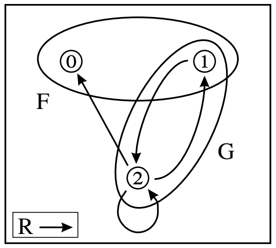
3 Structures are now the appropriate counterexamples to claims of validity. To build a structure that divides a dead-end gap, we take the alias sets of the gap and choose a range that contains a value corresponding to each alias set. Then we assign extensions to unanalyzed terms and functors so that the reference value each compound term will be the value corresponding to the term's alias set. Finally, we assign extensions to predicates by seeing what terms the resources affirm or deny these predicates of. Our new rules for closing gaps ensure that these instructions are consistent and that a structure built in this way will divide the dead-end gap. Such a structure can also be found as at least a part of a possible world.

Glen Helman 03 Aug 2010

6.4.x. Exercise questions

1. Each of **a**, **b**, and **c** gives a structure in one of the two sorts of presentation described in this section—by a diagram or by tables. Present each of them in the other way.

a.



b.

τ	Fτ
0	T
1	T
2	F

τ	Gτ
0	F
1	F
2	T

R	0	1	2
0	T	T	T
1	F	T	F
2	F	T	T

c.

τ	Fτ
0	T
1	T
2	F

τ	Gτ
0	F
1	T
2	T

τ	Hτ
0	T
1	F
2	T

R	0	1	2
0	F	T	F
1	T	F	F
2	F	T	F

2. Calculate a truth value for each of the following sentences on the structure used as the chief example in this section (see, for example, Figure 6.4.2-7):

a. (Fa ∨ Gb) → Rab

b. R(fca)(fac)

c. fab = fba

3. Use derivations to check each of the claims below; if a claim of entailment fails, use either tables or a diagram to present a structure that divides an open gap.

a. a = a → Fa ⊨ Fa

b. ¬ (Fa ∧ Fb) ⊨ ¬ Fa → ¬ Fb

c. a = b ∨ b = a ⊨ a = b ∧ b = a

d. Fa → a = b, ga = b, Ra(ga) → Fa, F(ga) ⊨ Raa → R(ga)(ga)

e. a = b → Rac, ¬ a = b → Rbc ⊨ Rbc

For more exercises, use the exercise machine.

6.4.xa. Exercise answers

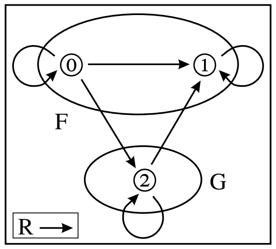
1. a.

τ	Fτ
0	T
1	T
2	F

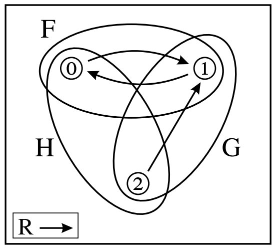
τ	Gτ
0	F
1	T
2	T

R	0	1	2
0	F	F	F
1	F	F	T
2	T	T	T

b.



c.



2. a. $\dfrac{\text{(Fa} \lor \text{Gb)} \rightarrow \text{Rab}}{\text{F3 T T4} \quad Ⓕ \quad \text{F34}}$ **b.** $\dfrac{\text{R(fca)(fac)}}{Ⓣ \quad \text{103} \quad \text{430}}$ **c.** $\dfrac{\text{fab = fba}}{\text{034 } Ⓕ \text{ 243}}$

3. a. *Without attachment rules:* *Using attachment rules:*

```
        │ a = a → Fa    2
        ├─
        │ │ ¬ Fa         (2)
        │ ├─
2 MTT   │ │ ¬ a = a      (3)
        │ │ ●
        │ ├─
3 DC    │ │ ⊥            1
        ├─
1 IP    │ Fa
```

```
        │ a = a → Fa    2
        ├─
1 CE    │ a = a         X,(2)
2 MPP   │ Fa            (3)
        │ ●
        ├─
3 QED   │ Fa
```

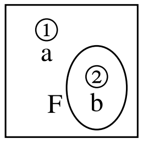
b.

```
          | ¬ (Fa ∧ Fb)      3
          |—
          | | ¬ Fa
          | |—
          | | | Fb           (3)
          | | |—
3 MPT     | | | ¬ Fa
          | | | ○            ¬ Fa,Fb ⊭ ⊥
          | | |—
          | | | ⊥            2
2 RAA     | | ¬ Fb           1
1 CP      | ¬ Fa → ¬ Fb
```

range: 1, 2

a	b
1	2

τ	Fτ
1	F
2	T



$$\frac{\neg\ (\text{Fa} \land \text{Fb})\ /\ \neg\ \text{Fa} \to \neg\ \text{Fb}}{\text{Ⓣ}\ \ \text{F1 F T2}\qquad \text{T F1 Ⓕ F T2}}$$

c.

```
          | a = b ∨ b = a      1
          | | a = b            a—b
          | |—
          | | | ●
          | | |—
3 EC      | | | a = b          2
          | | | ●
          | | |—
4 EC      | | | b = a          2
          | |—
2 Cnj     | | a = b ∧ b = a    1
          | | b = a            a—b
          | |—
          | | | ●
          | | |—
6 EC      | | | a = b          5
          | | | ●
          | | |—
7 EC      | | | b = a          5
          | |—
5 Cnj     | | a = b ∧ b = a    1
          |—
1 PC      | a = b ∧ b = a
```

d.

```
          | Fa → a = b            3
          | ga = b                a, b—ga
          | Ra(ga) → Fa           5
          | F(ga)
          |—
          | | Raa                 (6)
          | |—
          | | | ¬ R(ga)(ga)       (6)
          | | |—
          | | | | ¬ Fa            (5)
          | | | |—
5 MTT     | | | | ¬ Ra(ga)
          | | | | ○               b=ga,F(ga),Raa,¬ R(ga)(ga),¬ Fa,¬ Ra(ga) ⊭ ⊥
          | | | |—
          | | | | ⊥             4
4 IP      | | | Fa              3
          | | | | a = b         a—b—ga
          | | | | ●
          | | | |—
6 Nc=     | | | | ⊥             3
          | | |—
3 RC      | | | ⊥               2
          | |—
2 IP      | | R(ga)(ga)         1
          |—
1 CP      | Raa → R(ga)(ga)
```

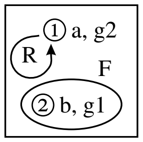
range: 1, 2

a	b
1	2

τ	gτ
1	2
2	1

τ	Fτ
1	F
2	T

R	1	2
1	T	F
2	F	F



$$\frac{\text{Fa} \to \text{a} = \text{b},\ \text{ga} = \text{b},\ \text{Ra(ga)} \to \text{Fa},\ \text{F(ga)}\ /\ \text{Raa} \to \text{R(ga)(ga)}}{\text{F1 Ⓣ 1 F 2}\ \ \text{2 1 Ⓣ 2}\ \ \text{F1 21}\ \ \text{Ⓣ F1 Ⓣ 21}\quad \text{T11 Ⓕ F 21 21}}$$

e.

```
          | a = b → Rac       3
          | ¬ a = b → Rbc     2
          |—
          | | ¬ Rbc           (2),(4)
          | |—
2 MTT     | | a = b           a—b, c; (3)
3 MPP     | | Rac             (4)
          | | ●
          | |—
4 Nc=     | | ⊥               1
          |—
1 IP      | Rbc
```

Glen Helman 03 Aug 2010