# Summary of First-Order Logic

1. **Sentential Logic**
   - Sentences and Connectives ............................................................... 2
   - Declarative Sentences and Truth-Values ........................................ 3
   - Truth-Functional Connectives ......................................................... 4

2. **Predicate Logic**
   - Introduction ....................................................................................... 5
   - Noun Phrases ....................................................................................... 5
   - Predicates ............................................................................................. 6
   - Quantifiers as Noun Phrases ............................................................ 7
   - A Problem with the Subject-Predicate Analysis of Quantifier Phrases ........................................ 8
   - Quantifier Phrases as Sentential Adverbs ........................................ 9
   - The Corresponding Analysis of ‘Some’ ........................................... 10
   - Scope .................................................................................................... 11
   - Quantifier Scope .............................................................................. 12

3. **Function Logic**
   - Compound Noun Phrases .................................................................. 13
   - Function-Signs .................................................................................. 14

4. **Identity Logic**
   - Introduction ....................................................................................... 16
   - The ‘is’ of Identity ........................................................................... 16
   - Numerical Quantifiers ....................................................................... 18

5. **Description Logic**
   - Subnectives ....................................................................................... 19
   - Definite Descriptions ....................................................................... 21
   - Russell's Theory of Descriptions ...................................................... 22
   - Free Logic .......................................................................................... 23
   - Proper Nouns in Free Logic .............................................................. 25
   - Descriptions in Free Logic ............................................................... 26
   - Scope ................................................................................................ 27
   - Scoped-Terms ................................................................................... 28
1. **Sentential Logic**

1. **Sentences and Connectives**

Sentential logic (SL) analyzes sentences in terms of just two grammatical categories:

```
  sentences
  connectives
```

Whereas sentences are a primitive category, connectives are a derivative category, the basic definition being given as follows.

\[(d1)\] A **connective** is an expression with blanks such that, filling these blanks with sentences results in a sentence.

Every connective has a *degree*, which is a natural number (0, 1, 2, ...); in particular, when the degree of a connective is \(k\), we say that it is a \(k\)-place connective. Officially:

\[(d2)\] Where \(k\) is any natural number (0, 1, 2, ...), A \(k\)-place connective is a connective with \(k\)-many blanks (places).

Connectives are a species of the genus *functor*, which is generally defined as follows.

\[(d3)\] A **functor** is an expression with blanks such that, filling these blanks with expressions of specified categories results in an expression of a specified category.

Once we have the general idea of a functor, we can categorize them according to their syntactic behavior. For example, a one-place connective takes one sentence and produces a sentence, which can be represented as follows.

\[S \rightarrow S\]

Similarly, a two-place connective takes two sentences and produces a sentence, which can be depicted as follows.

\[ (S \times S) \rightarrow S\]

This notation can also be simplified as follows.

\[S^2 \rightarrow S\]

More generally, a \(k\)-place connective is depicted as follows.\(^2\)

\[S^k \rightarrow S\]

---

\(^1\) 0-place connectives are a technical nicety that do not occur naturally. The most prominent example of such a connective is the contradiction symbol \(\perp\) used primarily in derivations. Note, however that \(\perp\) has other theoretical uses; for example, one can prove that all truth-functional connectives are definable in terms of \(\rightarrow\) and \(\perp\).

\(^2\) These are all the *simple functors* available in sentential logic. However, from an abstract categorial point of view, there are non-simple functors as well, i.e., functors that take functors as input. For example, a functor of category \[[(S \rightarrow S) \rightarrow S]\] takes a one-place connective as input and generates a sentence as output. The present author is unable to think of an example from ordinary language of such a functor. Nevertheless, the category is there in case we need it! For a general presentation of these ideas, please refer to the appendix on categorial grammar.
2. Declarative Sentences and Truth-Values

Ordinary sentential logic is not concerned with the whole class of sentences, but only *declarative sentences*, thus ignoring interrogative, imperative, exclamatory, and performative sentences. The simplest definition of a declarative sentence is that it is a sentence that is capable of being true or false. Basically, a declarative sentence is intended, when uttered, to declare something,\(^3\) which in turn is either true or false.

Associated with the adjectives ‘true’ and ‘false’ are the abstract proper nouns ‘True’ and ‘False’, which refer to what are known as *truth-values*.

An analogy might be useful here. Consider the difference between the adjective ‘blue’ and the proper noun ‘Blue’, as used in the following two sentences.\(^4\)

\[
\text{my favorite shirt is blue} \\
\text{my favorite color is Blue}
\]

We can readily hear the difference between these two sentences as soon as we invert them, as follows.

\[
\text{blue is my favorite shirt} \\
\text{Blue is my favorite color}
\]

The first one sounds funny (poetic, if you like); the second one sounds rather ordinary (prosaic, if you like).

Notice also that there is a correspondence between the adjective ‘blue’ and the noun ‘Blue’, given as follows.

\[
\text{object } x \text{ is blue} \\
\text{if and only if} \\
\text{the color of object } x \text{ is Blue [or: Blue is the color of object } x]\]

Now back to truth-values. There is an analogous relation in logic between the adjectives ‘true’ and ‘false’, on the one hand, and the abstract proper nouns ‘True’ and ‘False’.

\[
\text{sentence } S \text{ is true/false} \\
\text{if and only if} \\
\text{the truth-value of } S \text{ is True/False}
\]

---

\(^3\) The "something" that a sentence declares or expresses is usually referred to as a *proposition*.

\(^4\) Notice that we capitalize the proper noun to distinguish it from the corresponding adjective.

\(^5\) Note carefully the distinction between the 'is' of predication and the 'is' of identity. This sentence is symbolized in identity logic as follows. \(\forall x \{Bx \leftrightarrow [e(x)=B]\} \)
3. Truth-Functional Connectives

Ordinary SL does not deal with sentences in general, but only with declarative sentences. Furthermore, ordinary SL does not deal with all sentence connectives, but only with a special sub-class, known as truth-functional connectives. The following is the general definition.

\[(d4)\] A connective is **truth-functional** if and only if the truth-value of any compound formula constructed using that connective is a function of the truth-values of the constituent parts.

To be somewhat more specific, the following is the definition of truth-functionality as it applies to two-place connectives.

\[(d5)\] A two-place connective \(\odot\) is **truth-functional** if and only if the following obtains: for any sentences \(S_1\) and \(S_2\), the truth-value of \((S_1 \odot S_2)\) is a function of the respective truth-values of \(S_1\) and \(S_2\).

Now, the function referred to in this definition is called the **truth-function associated with** connective \(\odot\). Note carefully that we use the very same symbol both for the connective and for its associated truth-function; see below for examples.

Traditionally, there are five truth-functional connectives studied by SL.

<table>
<thead>
<tr>
<th>Connective</th>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>negation</td>
<td>(\neg)</td>
<td>not…</td>
</tr>
<tr>
<td>conjunction</td>
<td>&amp;</td>
<td>…and…</td>
</tr>
<tr>
<td>disjunction</td>
<td>(\vee)</td>
<td>…or…</td>
</tr>
<tr>
<td>conditional</td>
<td>(\rightarrow)</td>
<td>if…then…</td>
</tr>
<tr>
<td>biconditional</td>
<td>(\leftrightarrow)</td>
<td>…if and only if…</td>
</tr>
</tbody>
</table>

Whereas negation is a one-place connective that is written in prefix notation, the remaining connectives are two-place connectives, and are written in infix notation.

These are not all the truth-functional connectives; indeed, there are infinitely many of these. On the other hand, all truth-functional connectives are definable in terms of the above connectives; in fact, all truth-functional connectives can be defined in terms of just ‘\(\neg\)’ and ‘\(\&\)’.

Being truth-functional, each of the above connectives has associated with it a truth-function, which we denote by the very same symbol. These functions are depicted in the following tables [where the truth-values are abbreviated in the customary manner by ‘T’ and ‘F’].

<table>
<thead>
<tr>
<th>Truth-Value</th>
<th>(T)</th>
<th>(F)</th>
<th>(T)</th>
<th>(F)</th>
<th>(T)</th>
<th>(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\neg T)</td>
<td>(\neg T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
</tbody>
</table>
2. **Predicate Logic**

1. **Introduction**

The move from sentential logic to predicate logic involves three major new concepts.

(1) noun phrases  
(2) predicates  
(3) quantifiers

2. **Noun Phrases**

The first item represents a new primitive category – which we call "noun phrases", and which we abbreviate by ‘N’. As we propose to use the term, a noun phrase is basically an expression that can serve as a subject or object of a verb. The following are all examples of noun phrases.

<table>
<thead>
<tr>
<th>category</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>proper nouns</td>
<td>Mozart, Haydn, Jupiter, Saturn, The Eiffel Tower</td>
</tr>
<tr>
<td>pronouns</td>
<td>I, you, he, she, it, they</td>
</tr>
<tr>
<td>compound noun phrases</td>
<td>Jay’s mother, 2+2, the square root of 2</td>
</tr>
<tr>
<td>descriptive noun phrases</td>
<td>the man standing next to Bill, the three tallest buildings</td>
</tr>
<tr>
<td>nominalized verbs</td>
<td>running, to run, as in, I like to run</td>
</tr>
<tr>
<td>direct quotations</td>
<td>“snow is white”, as in, Jay said “snow is white”</td>
</tr>
<tr>
<td>indirect quotations</td>
<td>that snow is white, as in, Jay said that snow is white</td>
</tr>
</tbody>
</table>

Noun phrases come with a *number*, which is either singular or plural. In the case of pronouns, this is drilled into us in middle school, if not before. For example, we know that the pronoun ‘we’ is first person *plural*, and we know that the pronoun ‘she’ is third person *singular*. In the case of other such expressions, perhaps the easiest way to decide number is to ask what the appropriate form of the verb ‘to be’ is — if the appropriate verb is ‘is’, then the noun phrase is singular; if the appropriate verb is ‘are’, then the noun phrase is plural; simple as that! And pronouns mostly work the same way; I leave the reader to think about what the exceptions are.

In classical predicate logic, only singular noun phrases – officially called *singular terms* – are considered, whereas plural noun phrases are either ignored or finessed. Also, a noun phrase can be grammatically simple or complex, but predicate logic completely ignores their internal structure. For example, from the viewpoint of predicate logic, all of the following are atomic (simple).

- Jay’s mother
- 5+7
- the woman standing near the window

---

6 Grammatical complexity is related to, but not identical to, lexical complexity. For example, ‘the Eiffel Tower’ consists of three words, but it is regarded as grammatically simple.
3. Predicates

In addition to noun phrases, predicate logic concerns predicates. Whereas the category S of sentences, and the category N of noun phrases, are primitive categories, the category of predicates is a derivative (functor) category, just like connectives. The notion of predicate is defined as follows, which is followed by the subordinate definition of k-place predicate.

(d1) A predicate is an expression with zero or more blanks such that, filling these blanks with noun phrases results in a sentence.

(d2) Where k is any natural number (0, 1, 2, ...), a k-place predicate is a predicate with k places (blanks).

Notice that we allow k to be 0. Grammatically, a 0-place predicate is a predicate that takes no grammatical subject. The best examples of subject-less declarative sentences occur in weather reports – for example, ‘it is raining’, ‘it is snowing’, etc. Even though these sentences have an official grammatical subject ‘it’, it is obvious that ‘it’ does not refer to anything. What exactly is "it" in ‘it is raining’? Since there is no "real" subject in such sentences, when we symbolize them in predicate logic (see below), the ‘it’ simply disappears.

Next, if we want to depict the various categories of predicates, we do so as follows.

\[ N^0 \to S \quad \text{0-place predicates} \]
\[ N^1 \to S \quad \text{1-place predicates} \]
\[ N^2 \to S \quad \text{2-place predicates} \]

etc.

The standard symbolization technique for predicates follows a general pattern, given as follows.

Predicates are symbolized by upper case Roman letters.
Singular terms are symbolized by lower case Roman letters.\(^8\)
The predicate is written first, followed by its arguments.

**Examples**

<table>
<thead>
<tr>
<th>subject</th>
<th>∅</th>
<th>it</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicate</td>
<td>R</td>
<td>is raining</td>
</tr>
<tr>
<td>sentence</td>
<td>R</td>
<td>is raining</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>subject</th>
<th>J</th>
<th>Jay</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicate</td>
<td>T</td>
<td>is tall</td>
</tr>
<tr>
<td>sentence</td>
<td>T J</td>
<td>Jay is tall</td>
</tr>
</tbody>
</table>

---

\(^7\) Compare this situation to Italian in which, to say that it is raining, one simply says ‘piove’.  
\(^8\) More specifically, we propose to use ordinary lower case letters for constants and variables, and to use small caps for proper nouns, so for example, ‘K’ abbreviates ‘Kay’. See Chapter “General First-Order Logic” for the logical difference between constants and proper nouns.
This is the minimal scheme. The maximal scheme has the following additional features.

**Maximal punctuation scheme:**
Predicates come with square brackets, and explicitly marked blanks.
Arguments are separated by commas.

This adjusts the above translations as follows.\(^9\)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>R</td>
<td>K</td>
<td>RJK</td>
</tr>
<tr>
<td>Jay</td>
<td>respects</td>
<td>Kay</td>
<td>Jay respects Kay</td>
</tr>
<tr>
<td>J</td>
<td>S</td>
<td>K</td>
<td>RJKL</td>
</tr>
<tr>
<td>Jay</td>
<td>recommended</td>
<td>Kay</td>
<td>Jay recommended Kay to Elle</td>
</tr>
</tbody>
</table>

| \(\ldots\) is tall | \(T[\alpha]\) | \(\alpha\) is tall |
| \(\ldots\) respects \(\ldots\) | \(R[\alpha,\beta]\) | \(\alpha\) respects \(\beta\) |
| \(\ldots\) recommended \(\ldots\) to \(\ldots\) | \(R[\alpha,\beta,\gamma]\) | \(\alpha\) recommended \(\beta\) to \(\gamma\) |

| Jay is tall | \(T[J]\) |
| Jay respects Kay | \(R[J,K]\) |
| Jay recommended Kay to Elle | \(R[J,K,L]\) |

### 4. Quantifiers as Noun Phrases

Predicate Logic involves predicates, but more importantly it involves quantification. In particular, all logical inferences in predicate logic are performed in reference to the two special quantifiers – \(\forall\) (universal) and \(\exists\) (existential). For this reason, predicate logic is also called quantifier logic.\(^{10}\)

To understand quantification, one must begin by recognizing that, in ordinary language, quantifier phrases are noun phrases. For example, the sentences

all humans are mortal
every human is mortal

have the following grammatical form.

---

\(^9\) The Greek letters are schematic place holders, which serves as labeled blanks.

\(^{10}\) Quantifier logic also includes the logic of function-signs, which we examine in a later chapter.
Traditional logic hypothesizes that the *grammatical* form is also the *logical* form. Accordingly, the following all have the same logical form.

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>all humans</td>
<td>are mortal</td>
</tr>
<tr>
<td>every human</td>
<td>is mortal</td>
</tr>
</tbody>
</table>

Jay is mortal  
Kay is mortal  
every human is mortal

In particular that form may be represented as follows.

predicate[subject]

We have already seen that predicate logic symbolizes the first two sentences as follows

\[
\begin{align*}
M[J] \\
M[K]
\end{align*}
\]

If ‘every human is mortal’ has the same *logical* form, then its translation should be something like:

M[every human]

If we abbreviate ‘every’ by ‘\( \forall \)’ and ‘human’ by ‘\( H \)’, then the symbolization looks thus.

\[M[\forall H]\]

5. **A Problem with the Subject-Predicate Analysis of Quantifier Phrases**

Logic starts with grammar, but it does not end there! How do we formally analyze the following sentences?

1. every human is *not* mortal
2. every human is *im*mortal
3. *not* every human is mortal

In particular, do they have the same respective forms as the following.

Jay is *not* mortal  
Jay is *immortal*  
*not* Jay is mortal\(^{11}\)

The latter are all equivalent, and are straightforwardly symbolized as:

\[\sim M[J]\]

So, if the quantified sentences have the same forms as their unquantified counterparts, then they are all symbolized as follows.

\[\sim M[\forall H]\]

\(^{11}\) Prefixing ‘not’ does not generally produce a grammatical English sentence, unless we understand that, *at the beginning of a sentence*, ‘not’ is officially pronounced “it is not true that”.

The problem with this analysis is that the original sentences (1)-(3) are not semantically equivalent, so it would be a very bad idea to translate them all as a single logical formula. Clearly, (2) and (3) are not equivalent. What about (1)? Well, it is ambiguous, since two different grammatical constructions lead to the same surface expression. According to one construction, ‘not’ modifies the verb ‘is’ and negates the whole sentence. According to the other construction, ‘not’ modifies the adjective ‘mortal’ to form a new adjective ‘not-mortal’. In other words, sentence (1) is ambiguous in meaning between sentence (2) and sentence (3).

The moral of this story for ordinary users of ordinary English is quite simple — never use sentences like (1), but rather use sentences like (2) or like (3), depending upon which one is intended. The moral of this story for logical analysis is somewhat different. Our initial analysis of the two sentences

\[
\text{every human is \textbf{immortal}} \\
\text{not every human is mortal}
\]

produces the very same formula.

\[
\sim M[\forall H]
\]

This formula is just as loathsome as the unfortunate sentence ‘every human is not mortal’; it is ambiguous. For example, on one reading of ‘not’, the argument form

\[
\sim M[\forall H] \quad \sim M[j]
\]

is valid, but on the other reading of ‘not’, it is invalid. Logically speaking, this is a disaster!

### 6. Quantifier Phrases as Sentential Adverbs

We are now in position to reconstruct the key insight of modern symbolic logic. What distinguishes modern symbolic logic from all logic that preceded it is that quantifier phrases are treated, not as noun phrases, but as sentential adverbs, which are a species of sentential operator (connective). Turning quantifiers into sentential operators does not necessarily make for better grammar, but it does make for better logic, in the sense that it makes the principles of reasoning much easier to formulate.

In order to understand how the transformation takes place, we begin with our earlier example.

\[
\text{every human is mortal}
\]

Next, by a transformation known as quantifier-movement,\(^\text{12}\) we move the subject phrase from its original grammatical position forward in the sentence to produce the following sentence.

\[
\text{every human is such that he/she is mortal.} \\
\text{or: it is true of every human that he/she is mortal}
\]

Next, we propose the following logico/grammatical analysis.

\(^\text{12}\) This is exactly on a par with the transformation called wh-movement (‘who’ ‘what’ ‘how’ etc.), which is probably the most famous grammatical transformation, originally discovered by Noam Chomsky.
<table>
<thead>
<tr>
<th>sentential-adverb phrase</th>
<th>every human is such that…</th>
<th>(\forall H/x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>it is true of every human that…</td>
<td>(\forall H/x)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input sentence</th>
<th>he/she is mortal</th>
<th>M[x]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>resulting sentence</th>
<th>every human is such that he/she is mortal</th>
<th>it is true of every human that he/she is mortal</th>
<th>(\forall H/x) M[x]</th>
</tr>
</thead>
</table>

Note the new item – the variable ‘x’ – which performs a number of inter-related grammatical functions.

1. it marks the original grammatical location of the quantifier phrase;
2. it symbolizes the third person singular pronoun ‘he/she’;
3. it marks the quantifier phrase ‘every human’ [\forall H] as the pronominal antecedent of the pronoun ‘he/she’.

Although linguists prefer to stop their analysis with the formula

(\forall H/x) M[x]  every human x is such that x is mortal

logicians propose one more transformation, which reduces specific universal quantifier phrases

\textit{every human}, \textit{every apple}, \textit{every electron}, etc.,

to a single generic universal quantifier:

\textit{every thing}

This is accomplished by the following paraphrase.

\textit{every A is such that it is B}

is paraphrased as:

\textit{every thing is such that, IF it is A, THEN it is B}

The generic quantifier expression

\textit{every thing is such that …

it is true of every thing that …}

is symbolized by ‘\forall’ together with an index variable – for example, ‘\forall x’ or ‘\forall y’ or ‘\forall z’.

Applying this procedure to our original sentence

\textit{every human is mortal,}

we obtain the following symbolization.

\( \forall x(Hx \to Mx) \)

\textit{every thing is such that: if it is human, then it is mortal}
7. The Corresponding Analysis of ‘Some’

A similar story can be told about sentences involving ‘some’. Let us start with the following colloquial sentences.

some humans are mortal
some human is mortal

These have the following grammatical form.

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>some humans</td>
<td>are mortal</td>
</tr>
<tr>
<td>some human</td>
<td>is mortal</td>
</tr>
</tbody>
</table>

Symbolizing these in simplified predicate logic, we obtain the following common formula.

\[ M[\exists H] \]

Next, we apply quantifier-movement, which moves the quantifier phrase forward in the sentence to produce the following sentence.\(^\text{13}\)

some human is such that he/she is mortal
it is true of at least one human that he/she is mortal

Here, we understand the grammatical form as:

<table>
<thead>
<tr>
<th>sentential-adverb phrase</th>
<th>some human is such that…</th>
<th>(( \exists H/x ))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>it is true of some human that…</td>
<td></td>
</tr>
<tr>
<td>input sentence</td>
<td>he/she is mortal</td>
<td>M[x]</td>
</tr>
<tr>
<td>resulting sentence</td>
<td>some human is such that he/she is mortal</td>
<td>(( \exists H/x )) M[x]</td>
</tr>
<tr>
<td></td>
<td>it is true of some human that he/she is mortal</td>
<td></td>
</tr>
</tbody>
</table>

Finally, we replace the specific quantifier ‘some human’ by a generic quantifier ‘some thing’, by way of the following paraphrase.

some \( A \) is such that it is \( B \)
something is such that, it is \( A \), and it is \( B \)
or:
there is something such that, it is \( A \) and it is \( B \)

The quantifier expression ‘something is such that...’ is symbolized by ‘\( \exists \)’ together with an index variable – for example: \( \exists x \). This yields the following symbolization.

\[ \exists x(\H x & M x) \]

\(^{13}\) In keeping with first-order methodology, we ignore the plural form ‘some humans’ and concentrate on the singular form ‘some human’. We discuss the plural form in a later chapter concerning second-order logic.
8. Scope

Treating quantifiers as sentential adverbs allows us to construct formulas that are considerably more complex than any formula that is written in subject-verb-object form. It also allows us to construct a powerful, but simple, set of inference rules, as we see later. Equally importantly from a grammatical viewpoint, treating quantifiers as sentential adverbs endows them with a new logico-grammatical feature – scope.

Scope, which is an algebraic notion, is a prominent feature of both symbolic logic and symbolic arithmetic. For example, the following arithmetical expressions,

(e1) two plus three times four
(e2) the square root of four plus five
(e3) minus two squared

are all ambiguous. For example, (e1) is ambiguous between the following.

(i1) two plus three … times four
(i2) two plus … three times four

The logical pause represented by ‘…’ tells us which operator is the major operator – i.e., which operator has wide scope.

Now, as everyone knows, the official algebraic method of indicating scope is to use parentheses, which are prominent in both sentential logic and arithmetic. For example, the two informal expressions (i1) and (i2) can be replaced by the following algebraic expressions, respectively.

(a1) \((2 + 3) \times 4\) \([ = 20]\)
(a2) \(2 + (3 \times 4)\) \([ = 14]\)

Similarly, (e2) is ambiguous between the following.

the square root of four … plus five \(sr(4) + 5\) \([ = 7]\)
the square root of … four plus five \(sr(4 + 5)\) \([ = 3]\)

Finally, (e3) is ambiguous between the following.

minus two … squared \((-2)^2\) \([ = 4]\)
minus … two squared \((-2^2)\) \([ = -4]\)

Sentential examples follow a similar pattern. For example, the following expressions are ambiguous.

not \(A\) and \(B\)
not \(A\) … and \(B\) \(\sim A \& B\)
not … \(A\) and \(B\) \(\sim (A \& B)\)
\(A\) and \(B\) or \(C\)
\(A\) and \(B\) … or \(C\) \((A \& B) \lor C\)
\(A\) and … \(B\) or \(C\) \(A \& (B \lor C)\)

\[14\] These are examples of harmful ambiguity, which are distinguished from expressions like ‘two plus three plus four’, which are ambiguous, but not harmfully so. Also note that numerous conventions can be adopted according to which expressions without needed parentheses have preferred parses. For example, some calculators will parse the input string ‘\(2 + 3 \times 4 = \)’ by adding 2 and 3, and then multiplying the result by 4. Other calculators treat \(\times\) as out-scoping + by default, and accordingly add 2 to the result of multiplying 3 by 4.
9. **Quantifier Scope**

Let us now return to the formula

\(~M[\forall H]\)

which corresponds roughly to the sentence

every human is *not* mortal

When we proceed to transform the quantifier phrase ‘every human’ into a sentential adverb, we notice immediately that there are two equally plausible ways it can be moved forward in the formula. It can be moved in front of the ‘\(M\)’, or it can be moved in front of the ‘\(~\)’, which results in the following two formulas, respectively.

\((m1) \quad \sim (\forall H/x) \ M[x]\)
\((m2) \quad (\forall H/x) \sim M[x]\)

In the first case, the quantifier is given *narrow scope* relative to the negation operator; in the second case, the quantifier is given wide *scope relative* to the negation operator.

The difference between these two becomes more obvious perhaps when we transform the specific quantifier ‘every H’ into a generic quantifier, as follows.

\((g1) \quad \sim \forall x(\text{H}x \rightarrow \text{M}x)\)
\((g2) \quad \forall x(\text{H}x \rightarrow \sim \text{M}x)\)

These of course correspond to our original unambiguous sentences.

\((s1) \quad \text{not every human is mortal}\)
\((s2) \quad \text{every human is immunortal}\)

3. **Function Logic**

1. **Compound Noun Phrases**

If we symbolize the following sentence

Jay's mother is tall

in predicate logic, we obtain the following formula,

\[ Tm \]

alt: \[ T[m] \]

if we use a bracket convention for predicates. The following is our abbreviation scheme.

\[ T[\alpha] \quad \alpha \text{ is tall;} \]
\[ m \quad \text{Jay's mother.} \]

Similarly, if we symbolize

two plus three is greater than four \[ 2+3 > 4 \]
in predicate logic, we obtain

\[ Gpf \]

alt: \[ G[p, f] \]

where the following is the abbreviation scheme.

\[
\begin{align*}
G[\alpha, \beta] & \quad \alpha \text{ is greater than } \beta \\
p & \quad 2+3 \\
f & \quad 4
\end{align*}
\]

These translations are inadequate, because they do not uncover the additional structure contained in the complex noun phrases ‘Jay's mother’ and ‘2+3’. In order to accomplish this, we need to analyze these noun phrases into the noun phrases of which they are composed. In the case of ‘Jay's mother’, there is one component noun phrase – ‘Jay’. In the case of ‘2+3’, there are two component noun phrases – ‘2’ and ‘3’.

2. Function-Signs

Both of these expressions are analyzed logically in terms of the notion of function-sign, which is a new kind of functor, officially defined as follows.

(d3) A function-sign is an expression with (zero or more) blanks such that, filling these blanks with noun phrases results in a noun phrase.

As with our earlier functors – connectives and predicates – every function-sign has a degree. This is summarized as followed.

(d4) Where \( k \) is any natural number (0, 1, 2, ...), A \( k \)-place function-sign is a function-sign with \( k \) many blanks (places).

The following are the various categories of function-signs.

\[
\begin{align*}
N^0 & \rightarrow N & \text{0-place function-sign} \\
N^1 & \rightarrow N & \text{1-place function-sign} \\
N^2 & \rightarrow N & \text{2-place function-sign} \\
& \text{etc.}
\end{align*}
\]

Now, let us go back and examine our earlier examples. First, in

Jay's mother is tall

the subject is

Jay's mother.

This is a compound noun phrase, which consists of two grammatical components,

Jay

which is a proper noun, and hence atomic, and

___'s mother,
which is a one-place function-sign. If one fills the blank with a noun phrase, simple or complex, the result is also a noun phrase. It might be an absurd noun phrase, like ‘33’s mother’, but nonetheless it is syntactically speaking a noun phrase.

Now, the way we depict function-signs in function logic is very similar to the way we depict predicates. The difference is that we use lower case letters instead of upper case letters, and we use round parentheses instead of square brackets. Thus, for example:

\[ m(\alpha) \] stands for \( \alpha \)'s mother

Therefore, if ‘J’ stands for the proper noun ‘Jay’

\[ m(j) \] stands for Jay's mother

So, when we plug back in, we obtain our final formula.

\[ T[m(j)] \]

In our arithmetic example,

two plus three is greater than four \[ 2 + 3 > 4 \]

the two immediate noun phrases are ‘two plus three’ and ‘four’. Whereas the latter is a simple proper noun, the former is a complex noun phrase. In fact, it consists of three grammatical components:

two
three

which are proper nouns, and

___ plus ___

which is a 2-place function-sign, which we abbreviate as follows.

\[ p(\alpha, \beta) \] \( \alpha \) plus \( \beta \)

So, when we plug this back into our formula, we obtain:

\[ G[p(2, 3, 4)] \]

Notice that we use the Arabic numerals in our symbolization, mostly for convenience.
4. Identity Logic

1. Introduction

No logical analysis of mathematics, or ordinary language for that matter, would be complete without a logical analysis of such phrases as:

- two plus two is four
- two plus three is not four
- three plus four is seven
- etc.

In function logic, we already have the purely grammatical means to symbolize these, as follows.

\[
\begin{align*}
I[p(2, 2), 4] \\
\sim I[p(2, 3), 4] \\
I[p(3, 4), 7]
\end{align*}
\]

Lexicon:

\[
\begin{align*}
p(\alpha, \beta) & \quad \alpha \text{ plus } \beta \\
I[\alpha, \beta] & \quad \alpha \text{ is } \beta
\end{align*}
\]

Here, the verb ‘is’ is represented as a two-place predicate, not unlike ‘respects’. The problem, however, is not grammatical, but logical. The following is a valid argument.

\[
\begin{align*}
2+2 \text{ is } 4 & \quad I[p(2, 2), 4] \\
4 \text{ is even} & \quad E[4] \\
\text{therefore, } 2+2 \text{ is even} & \quad / E[p(2, 2)]
\end{align*}
\]

But its logical form, as depicted to the right, is not a valid form in function logic.

2. The ‘is’ of Identity

The problem is that ‘is’, as used in these sentences, is not just any 2-place predicate. It is a very special 2-place predicate, what we would indeed call a logical predicate. Our examples are from arithmetic, but ‘is’ is not peculiar to arithmetic. Arithmetic provides the easiest examples, since everyone is familiar with it. Other examples of this special two-place ‘is’ include the following.

- who is the person standing next to the window?
  that is Jay’s mother.

- who is the president of the U.S.?
  George Bush is the President of the U.S.

What these examples have in common is the logical predicate ‘is’, which is sometimes called:

the ‘is’ of identity

Notice that in asking someone to identify a person, we are asking “who is that person?” We are using the verb ‘is’ as the logical notion of identity.

This use of ‘is’ is in contrast to two other uses.
the ‘is’ of predication
the ‘is’ of existence

We are completely familiar with the ‘is’ of predication. If we say ‘Jay's mother is tall’, the word ‘is’ is part of the predicate; it is not grammatically autonomous in our analysis. The ‘is’ of existence is best exemplified by Hamlet's soliloquy “To be, or not to be; that is the question.” The ‘is’ of existence is also apparent in the expression ‘there is a thing such that ...’, which we refer to as the **existential quantifier**.

Back to the ‘is’ of identity. Well, we already know how it is symbolized in arithmetic – by the equals sign ‘=’. Logicians have preserved this venerable symbol, and have adopted it to stand for the ‘is’ of identity. The following is our official syntax.

if σ and τ are singular-terms, then \([σ = τ]\) is a formula.

Notice carefully: Whereas the run-of-the-mill, non-logical, predicates are always written in prefix notation, special, logical, predicates are written in "natural" notation, as in arithmetic. Notice also the appearance of the square brackets; this is a vestige of the bracket notation for predicates. In an important sense, they are completely optional, and are mostly used to help visually parse formulas in which ‘=’ appears. The following are examples.

\[\forall x\exists y[\bar{x} = \bar{y}]\]
\[\forall x\exists y[\bar{x} = \bar{y}]\]
\[\forall x\forall y[\bar{x} = \bar{y} \to (Fx \to Fy)]\]

Note that the brackets have been dropped in the third example, since they do not visually contribute to parsing.

Also note that the second example can be simplified as follows.

\[\forall x\exists y[\bar{x} \neq \bar{y}]\]

We regard the expression ‘\(x \neq y\)’ as mere shorthand for ‘\(\sim[\bar{x} = \bar{y}]\)’, for all purposes, including most importantly applying SL-rules like \(\forall O\) and \(\rightarrow O\). For example, the following is a valid application of \(\rightarrow O\).

\[Fx \to \bar{x} = \bar{y}\]
\[x \neq \bar{y}\]

\[\sim Fx\]

Of course, presenting ‘=’ as a logical predicate is incomplete without also presenting its associated inference rules. In this connection, there are four rules that are usually presented – Reflexivity, Symmetry, Transitivity, and Leibniz's Law.
Notice that R= is a zero-place rule. LL is simply the principle of substitution, familiar to anyone who has studied high school algebra. The following are examples.

<table>
<thead>
<tr>
<th>Jay's mother is tall</th>
<th>T[m(i)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jay's mother is the president</td>
<td>m(i) = p</td>
</tr>
<tr>
<td>the president is tall</td>
<td>T[p]</td>
</tr>
<tr>
<td>4 is even</td>
<td>E[4]</td>
</tr>
<tr>
<td>2+2 is 4</td>
<td>2+2 = 4</td>
</tr>
<tr>
<td>2+2 is even</td>
<td>E[2+2]</td>
</tr>
</tbody>
</table>

3. **Numerical Quantifiers**

Not only does the logic of identity help us do arithmetic, which is the science of natural numbers, it also helps us understand the numbers themselves – thought of as *quantities*.

From Intro Logic we already have at least one numerical concept – we can say zero in its most useful form. For example, we can say that there are zero-many unicorns, or that the number of unicorns is, i.e., equals (=), zero, by saying simply that there are no unicorns. The following is the translation from number-talk to quantifier-talk.

\[
\text{the number of unicorns} = 0 \quad \sim \exists x \text{U} x
\]

Similarly, since we can say zero, we can also say not-zero. For example, if the number of unicorns is not zero, then the number of unicorns is 1, or 2, or 3, or 4... (but not 3/4, or \(-7\), or \(\pi\)) In other words, the number of unicorns is *at least one*. Well, of course! We know how to say not-zero, which is the same as at-least-one; we have a quantifier just made for that – the existential quantifier. The following is the translation from number-talk to quantifier-talk.

\[
\begin{align*}
\text{the number of unicorns} &\neq 0 \\
\text{the number of unicorns} &> 0 \\
\text{the number of unicorns} &\geq 1
\end{align*}
\sim \exists x \text{U} x
\]

What about other numerical quantifiers? The following are examples.
there is exactly one unicorn
there is at most one unicorn
there are at least 2 unicorns
there are at most 2 unicorns
there are exactly 2 unicorns
etc.

How do we symbolize these? To make a long story short, the following are the translations of these sentences into the language of identity logic.

<table>
<thead>
<tr>
<th>#(U)=0</th>
<th>~∃xUx</th>
</tr>
</thead>
<tbody>
<tr>
<td>#(U)=1</td>
<td>∃x∀y(Uy ↔ y=x)</td>
</tr>
<tr>
<td>#(U)=2</td>
<td>∃x∃y(x≠y &amp; ∀z{Uz ↔ (z=x v z=y)})</td>
</tr>
<tr>
<td>#(U)=3</td>
<td>∃w∃x∃y(w≠x &amp; w≠y &amp; x≠y &amp; ∀z{Uz ↔ (z=w v z=x v z=y)})</td>
</tr>
<tr>
<td>#(U)≥1</td>
<td>∃xUx</td>
</tr>
<tr>
<td>#(U)≥2</td>
<td>∃x∃y(x≠y &amp; Ux &amp; Uy)</td>
</tr>
<tr>
<td>#(U)≥3</td>
<td>∃x∃y∃z(x≠y &amp; y≠z &amp; x≠z &amp; Ux &amp; Uy &amp; Uz)</td>
</tr>
<tr>
<td>#(U)≤1</td>
<td>∃x ∀y(Uy → y=x)</td>
</tr>
<tr>
<td>#(U)≤2</td>
<td>∃x∃y ∀z{Uz → (z=x v z=y)}</td>
</tr>
<tr>
<td>#(U)≤3</td>
<td>∃x∃y∃z ∀w{ Uw → (w=x v w=y v w=z)}</td>
</tr>
</tbody>
</table>

The logical analysis given above was first given by Russell, and provides the inspiration at the basis of Russell and Whitehead's epic *Principia Mathematica* (1910-1913). The analysis of 'exactly one' is the inspiration at the basis of Russell's outrageously famous theory of definite descriptions. We postpone examining this until we get to descriptions – which we will do shortly.

5. **Description Logic**

1. **Subnectives**

So far, we have talked about three categories of functors:

| connectives | $S^k \rightarrow S$ |
| predicates | $N^k \rightarrow S$ |
| function-signs | $N^k \rightarrow N$ |

Among these simple functor types, there is a notable asymmetry. By way of correcting this theoretical lapse, we introduce the fourth and final simple functor type.

subnectives $S^k \rightarrow N$

In other words, a subnective takes sentences as input and delivers a noun phrase as output.

Does ordinary language provide any examples of such functors. First, it is plausible to regard each of the following expressions as a one-place subnective.

---

Here, we understand the blanks are to filled by sentences. When so filled we have various noun phrases that can be used as subjects or objects of verbs, including the following.

<table>
<thead>
<tr>
<th>subject</th>
<th>verb (phrase)</th>
<th>direct object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socrates</td>
<td>said</td>
<td>that all humans are mortal</td>
</tr>
<tr>
<td>Socrates</td>
<td>said</td>
<td>“all humans are mortal”</td>
</tr>
<tr>
<td>Socrates</td>
<td>did not say</td>
<td>whether all humans are mortal</td>
</tr>
</tbody>
</table>

The first sentence is an example of what is usually called *indirect quotation*, according to which we quote Socrates by referring to the proposition he expressed, not by repeating his words. In this example, the direct object

that all humans are mortal

can be grammatically analyzed as follows.

<table>
<thead>
<tr>
<th>subnective</th>
<th>sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>that</td>
<td>all humans are mortal</td>
</tr>
</tbody>
</table>

This is in contrast to *direct quotation*, according to which we quote Socrates by repeating his very words. For example, it is highly unlikely that the following is literally true,

Socrates said “all humans are mortal”
because it is unlikely Socrates ever uttered this combination of words/sounds.16

Nevertheless, direct quotation affords us with another example of a subnective, in accordance with the following grammatical analysis.17

<table>
<thead>
<tr>
<th>subject</th>
<th>verb (phrase)</th>
<th>direct object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socrates</td>
<td>said</td>
<td>subnective</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“___”</td>
</tr>
</tbody>
</table>

The word ‘whether’ is syntactically similar to ‘that’, and so its analysis is left as an exercise.

---

16 There are, actually, different ways to repeat someone's "very" words, and so there are different ways to do direct quotation. Socrates may have said words (in Ancient Greek) that translate into English as these very words.

17 Direct quotation is notably deficient as a subnective, and should rather be called a *pseudo-subnective*. In particular, direct quotation is *completely opaque* semantically.
2. **Definite Descriptions**

Whether there are any simple subnectives is perhaps controversial.\(^\text{18}\) Description logic, however, does provide an example of a subnective-like functor (just as quantifiers are connective-like functors). The canonical ordinary language expression for the definite description is

\[
\text{the one/unique (person, thing, number, etc.) such that} \ldots
\]

which is symbolized:

\[\iota x\]

where ‘\(\iota\)’ is an upside down iota. Whether ‘\(\iota x\)’ (read ‘iota \(x\)’) means ‘the person...’, or ‘the thing...’, or ‘the number...’ will depend upon the domain of discourse.

The critical fact about ‘\(\iota\)’ is that, unlike the quantifier functors ‘\(\forall\)’ and ‘\(\exists\)’, ‘\(\iota\)’ is a subnective-like functor. To be specific, the following is its category.

\[(V_0 \times S) \rightarrow N\]

In other words, ‘\(\iota\)’ takes an individual variable, and a formula, as input, and yields a noun phrase as output. So the expression

\[\iota xF x\]

is a singular-term, not a formula. We read this expression as:

\[\text{the (unique) } x \text{ such that } F x\]

or:

\[\text{the one who is } F\]

or simply:

\[\text{the } F\]

In order to use this complex noun phrase to produce a formula, we need a predicate. The following are examples.

\[
\begin{align*}
T[\iota xF x] & \quad \text{the } F \text{ is } T \\
R[\iota xF x, \kappa] & \quad \text{the } F \text{ 's } \kappa \\
\iota xF x = \kappa & \quad \text{the } F \text{ is } \kappa
\end{align*}
\]

Notice that a description can appear inside another description. This is the basis of the nursery rhyme “This is the house that Jack built”. Consider the following sequence.

\[
\begin{align*}
t & = \iota x(Hx \& Bjx) \\
t & = \iota y\{My \& L[y, \iota x(Hx \& Bjx)]\} \\
t & = \iota z\{Cz \& C[z, \iota y\{My \& L[y, \iota x(Hx \& Bjx)]\}]\}
\end{align*}
\]

etc.

Here, the intended lexicon is as follows.

<table>
<thead>
<tr>
<th>(t)</th>
<th>this</th>
<th>(C[\alpha])</th>
<th>(\alpha) is a cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>(j)</td>
<td>Jack</td>
<td>(L[\alpha, \beta])</td>
<td>(\alpha) lived in (\beta)</td>
</tr>
<tr>
<td>(H[\alpha])</td>
<td>(\alpha) is a house</td>
<td>(C[\alpha, \beta])</td>
<td>(\alpha) chased (\beta)</td>
</tr>
</tbody>
</table>

---

\(^{18}\) By the way, how do you analyze this sentence – what is the subject?
3. **Russell's Theory of Descriptions**

There are three approaches to the logic of descriptions.

Frege's approach (classical first-order logic)
Russell's approach (analyze descriptions so they go away)
Free Logic approach (a compromise between the previous two)

We will discuss just the latter two.

According to Russell's theory of descriptions, any formula with a description, for example,

\[ \text{the } F \text{ is } G \]

must be *explicated/replaced* by an equivalent formula not explicitly containing the description. The following is Russell's proposed explicating formula.

\[ \text{there is exactly one } F, \text{ and it is } G \]

This can also be read

\[ \text{there is exactly one } F, \text{ which is } G \]

which should not be confused with

\[ \text{there is exactly one } F \text{ that is } G. \]

For the first one to be true, the number of F's must be 1. For the second one to be true, the number of F's may be any number, so long as exactly one of them is G.

We ignore the ‘that’ statement, and concentrate on the ‘which’ statement.

\[ \text{there is exactly one } F, \text{ which is } G \]

This can be translated into identity logic as follows.

\[ \exists x \{ \forall y( Fy \leftrightarrow y=x ) \land Gx \} \]

This can be read:

\[ \text{there is an } x \text{ such that } x \text{ is the only } F [x \text{ is } F, \text{ and nothing else is }] \land x \text{ is } G \]

Unfortunately, Russell's analysis faces an immediate problem. What happens when we try to explicate the following formula?

\[ \text{the } F \text{ is } \not G \]

According to Russell, there are two readings of this, according to how we logically locate the negation; there seem to be two choices as to the logical location of ‘not’.

---

the F is not-G  
it is not true that: the F is G 

The following are the corresponding Russell formulas. 

there is exactly one F, which is not G  
it is not true that: there is exactly one F, which is G 

And the following are the corresponding formulas. 

$$\exists x \{ \forall y(Fy \leftrightarrow y=x) & \sim Gx \}$$  
$$\sim \exists x \{ \forall y(Fy \leftrightarrow y=x) & Gx \}$$  

The important thing to realize is that these are **not logically equivalent**. This is regarded as an issue of scope. In one case, the scope of the description is wide; the other case, the scope of the description is narrow (both scopes in relation to ‘not’). In other words, the problem is no different from the problem of analyzing the following formula,  

every F is not H  

which has two readings. 

 every F is not-H  

it is not true that: every F is H  

We will get back to the issue of descriptive scope, when we talk about scoped-terms. 

### 4. Free Logic 

"Classical" first order logic, whose quantifier principles trace to Frege, is based on the following two presuppositions: 

(c1) The domain (universe) of discourse is not empty. Accordingly, the sentence ‘there is something’ is **logically** true, even though it is not **necessarily** true. 

(c2) Every singular-term – no matter how silly – denotes an existing object (i.e., an element of the domain). 

Free logic offers an alternative to classical logic that denies both (c1) and (c2). In order to formally implement free logic, we must adjust a number of inference rules. Consider the following dual-pair of rules of classical logic. 

\[ \begin{array}{|c|c|} 
\hline 
\forall O & \exists I \\
\hline 
\forall v \Phi & \Phi[v/x] \\
\Phi[v/x] & \exists v \Phi \\
\hline 
\end{array} \]

\[ 20 \text{ Technically, there are two different versions of free logic. The more radical version (Universally Free Logic) denies both (c1) and (c2). The less radical version of Free Logic rejects (c2), but accepts (c1). We are principally interested in the more radical version.} \]
Φ is any formula, ν is any variable, τ is any closed singular-term, and Φ[τ/ν] results from Φ when τ replaces every occurrence of ν that is free in Φ.

The main point is that, according to classical logic, any singular-term τ can be substituted for the variable ν. So, the following are examples.

\[
\begin{align*}
\forall x \text{H}x & \quad \text{H}[f(κ)] \\
\text{H}[m(μ)] & \quad \exists x \text{H}x
\end{align*}
\]

By contrast, free logic requires an additional premise for both ∀O and ∃I, as follows.

\[
\begin{array}{c|c}
∀O(f) & ∃I(f) \\
\hline
∀νΦ & Φ[τ/ν] \\
E![τ] & E![τ] \\
\hline
Φ[τ/ν] & ∃νF
\end{array}
\]

Here, the predicate ‘E!’ is the special logical predicate of "existence". If one has the resources of identity logic, then one can simply define existence as follows.

\[
\text{Def E!}
\begin{align*}
E![τ] & \quad \exists ν[ν = τ]
\end{align*}
\]

Here, τ is any closed singular-term, and ν is any variable.

The easiest way to implement free logic inside our natural deduction system is adjust the ∀O and ∃I rules as follows.

\[
\begin{array}{c|c}
∀O & ∃I \\
\hline
∀νΦ & Φ[o/ν] \\
\hline
Φ[o/ν] & ∃νF
\end{array}
\]

\(o\) must be an old constant, which is to say a constant that appears in a line that is neither boxed nor cancelled.

In order to reconstruct the original free logic quantifier forms, one must use the definition of ‘E!’, ∃O, and Leibniz's Law. The following schematic derivations show that the original free logic quantifier rules are admitted in our system of natural deduction.
5. Proper Nouns in Free Logic

This brings us to a fresh problem. In elementary predicate logic, we use constants (‘a’, ‘b’, etc.) in two quite different ways. First, they are used to abbreviate certain simple noun phrases (proper nouns). They are also used in derivations, in connection with UD and \( \bar{O} \). In classical logic, the distinction is unimportant. The reason is that, in classical logic, every singular-term, and hence every proper noun, automatically refers to an object in the domain.

In free logic, we can no longer afford this ambiguity. In free logic, whereas every variable and every constant (unquantified variable) refers to an element of the domain, proper nouns may be improper, which is to say they refer to nothing whatsoever.

Accordingly, in doing derivations in free logic, we must carefully distinguish between constants and proper nouns. This can be accomplished in a number of ways.

1. First Way

We can just be careful – specifically, by observing the following simple rule.

\[
(R!) \text{ Constants are exclusively derivational constructs, and only come into existence inside of derivations – in particular, in association with UD and } \exists O. 
\]

This rule has two corollaries – one logical, one practical.

\[
(c1) \text{ No lower case letter that is used in (the symbolization of the) argument to be proven by derivation is a constant; every such letter is a proper noun.} 
\]

\[
(c2) \text{ Do not choose as a constant any letter that serves as a proper noun!} 
\]

2. Second Way

We can adopt alternative notation. For example, we can use bold-face letters or small-caps for proper nouns, and use ordinary lower case letters for constants and variables.

3. Third Way

We can take advantage of our logical machinery, and put those kooky 0-place function-signs to work. In that case, we symbolize ‘Jay’ by ‘j()’ and ‘Kay’ by ‘k()’. 
4. **Fourth Way**

We can enlarge our theoretical syntax to include proper nouns as a primitive sort of expression; for pure logic, this is just like the second way. For applied logic, this is perfect for symbolizing theories that include explicit proper nouns; for example, arithmetic has (at least) the proper noun ‘0’.

We follow a combination of the first, second, and fourth ways. We have already seen symbolizations in which the native symbols are preserved ('0', ‘1’, etc.) We make this official now. We also do not have to rewrite any of our earlier derivations.

6. **Descriptions in Free Logic**

The treatment of descriptions in free logic is simply a special case of the treatment of all singular-terms [other than variables/ constants]. So, for example, the following arguments are not valid.

\[
\begin{array}{c|c}
\forall x Gx & G[\exists xFx] \\
G[\exists xFx] & \exists x Gx \\
\end{array}
\]

What is missing, in each case, is the premise.

\[
\begin{array}{c|c}
E[\exists xFx] & \exists x[x = 1xFx] \\
\end{array}
\]

Note that according to Frege's general account of singular-terms, both argument forms are valid, but according to Russell's account of descriptions, whereas the first one is invalid, the second one is valid. To see the latter, let us look at the derivation.

(1) \( G[\exists xFx] \) \( \text{Pr} \)
(2) \( \textbf{SHOW: } \exists x Gx \) \( \text{DD} \)
(3) \( \exists y \{ \forall x (Fx \leftrightarrow x=y) \land Gy \} \) \( 1, \text{Russell's analysis} \)
(4) \( \forall x (Fx \leftrightarrow x=a) \land Ga \) \( 3, \exists O \)
(5) \( Ga \) \( 4, \text{SL} \)
(6) \( \exists x Gx \) \( 5, \exists I \)

According to the free logic account of descriptions, as we present it, we require the additional premise that \( \exists xFx \) exists [or that the description ‘\( \exists xFx \)’ is proper]. Let us look at that derivation.

(1) \( G[\exists xFx] \) \( \text{Pr} \)
(2) \( E[\exists xFx] \) \( \text{Pr} \)
(3) \( \textbf{SHOW: } \exists x Gx \) \( \text{DD} \)
(4) \( \exists x[x = 1xFx] \) \( 2, \text{Def E!} \)
(5) \( a = 1xFx \) \( 4, \exists O \)
(6) \( Ga \) \( 1,5, \text{LL} \)
(7) \( \exists x Gx \) \( 6, \exists I \)

In free logic, we are not entitled to make Russell's move without the additional premise that \( \exists xFx \) exists. To see how this additional premise also yields the Russell formula, consider the following derivation.
The key line is (6), at which point the $\cdot \Omega$ rule is applied. The two $\cdot \Omega$ rules are given as follows.

<table>
<thead>
<tr>
<th>$\cdot \Omega$</th>
<th>$\cdot \Iota$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c = \nu \Phi$</td>
<td>$\forall \nu (\Phi \leftrightarrow \nu = c)$</td>
</tr>
<tr>
<td>$\forall \nu (\Phi \leftrightarrow \nu = c)$</td>
<td>$c = \nu \Phi$</td>
</tr>
</tbody>
</table>

$c$ must be a constant.

7. Scope

Scope is perhaps the single greatest contribution of modern logic to grammar. We see it in a couple of places in predicate logic. First, the sentence

everyone respects someone

and its passive counterpart

someone is respected by everyone

both have the same grammatical form, given as follows.

$R[\forall, \exists]$

When we go to transform this formula via quantifier-movement, we have a choice concerning which quantifier to do first, and which quantifier to do second. In particular, we obtain the following formulas.

$\forall x \exists y Rxy$

$\exists y \forall x Rxy$

The difference between these two formulas concerns which quantifier has wide scope, and which has narrow scope.

But the fact that the two admissible transformations are not equivalent strongly suggests that the original sentence is ambiguous. This can be clarified by saying that the original sentence is ambiguous between the following two readings.

everyone respects someone or other

everyone respects someone – the same someone

The latter can also be stated as follows.
there is someone ... whom everyone respects

Another place where scope issues arise is in sentences containing the quantifier ‘any’. We know that ‘any’ and ‘every’ are both universal quantifiers. In fact, we translate ‘everyone is H’ as ‘for any person x, x is H’. On the other hand, ‘any’ and ‘every’ are not interchangeable! Consider the following pair of sentences.

if everyone can fix your car, then I can
if anyone can fix your car, then I can

Clearly, these are not equivalent; the first one is trivially true; the second one is genuine bragging.

Also, consider the following pair of sentences.

Jay does not respect everyone
Jay does not respect anyone

Once again, these are not equivalent; the first one is plausibly true, no matter who Jay is; the second one is only true if Jay is a misanthrope.

Finally, consider the following pair of sentences.

no one respects everyone
no one respects anyone

Once again, these are not equivalent. The first one is plausibly true [you cannot find a single person who respects everyone]. The second one is in fact false, since it is easy to find at least one person who respects at least one person.

Now, the analysis of ‘any’ basically portrays it as a universal quantifier that usually comes attached to a correlative logical expression, with respect to which it (‘any’) has wide scope. This analysis works fine for ‘if any’ and ‘not any’; it does not work so well for ‘no...any’, which is best treated sui generis – as a whole new class of (double) quantifier.

So, for example, we have the following logical analyses.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Logical Form</th>
<th>Scope</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>if everyone is F, then so is Jay</td>
<td>$\forall x Fx \rightarrow F_j$</td>
<td>wide</td>
<td>$\forall$ is narrow</td>
</tr>
<tr>
<td>if anyone is F, then so is Jay</td>
<td>$\forall x (Fx \rightarrow F_j)$</td>
<td>wide</td>
<td>$\forall$ is narrow</td>
</tr>
<tr>
<td>Jay does not respect everyone</td>
<td>$\sim \forall x R_jx$</td>
<td>wide</td>
<td>$\forall$ is narrow</td>
</tr>
<tr>
<td>Jay does not respect anyone</td>
<td>$\forall x \sim R_jx$</td>
<td>wide</td>
<td>$\sim$ is narrow</td>
</tr>
</tbody>
</table>

8. Scoped-Terms

Frege taught us that quantifiers have scope. Russell taught us that certain singular-terms – i.e., descriptions – have scope. In order to correlate Russell's discovery with Frege's discovery, we must re-organize Russell's notation so as to parallel Frege's notation.

In the first-order logical analysis of quantified sentences, ordinary language quantifiers are split between two locations – the grammatical location, and the logical location. Whereas the former pertains to number, person, case, etc., the latter pertains exclusively to scope (logic's contribution!) In the logical analysis, whereas the grammatical position is depicted simply by a variable, the scope position is
depicted by the official logical quantifier expression itself (e.g., ‘∀x’). The latter, in turn, may be regarded as a special variable-binding sentential adverb (one-place connective). The transformational process, in its simplest instance, looks like the following.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H[∀]</td>
<td>everything is H</td>
</tr>
<tr>
<td>∀xH[x]</td>
<td>everything is such that it is H</td>
</tr>
</tbody>
</table>

The transformation does three things.

1. The quantifier is raised (in a style similar to Chomsky's ‘wh’-movement);
2. The "deep structure" location of the quantifier is marked by a variable;
3. This same variable is attached to the quantifier, for cross-referencing.

Now, if we wish to apply the transformational strategy to Russell's descriptions, we proceed pretty much the same way. In its simplest instance, it goes as follows.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H[1xFx]</td>
<td>the F is H</td>
</tr>
<tr>
<td>(1xFx/y) H[y]</td>
<td>the F is such that it is H</td>
</tr>
</tbody>
</table>

Once again, the transformation does three things – (1) description movement, (2) deep-structure marking, (3) cross referencing. Here, the cross-referencing variable is ‘y’, but it could also be ‘x’, except that ‘x’ might be visually confusing (though not to a computer!). Another visual device is utilized, since descriptions are themselves grammatically complex. We enclose the new sentential adverb in parentheses, and we separate the description from the cross-referencing variable. If we wanted our procedures to be consistent, then we might go back and rewrite quantifier expressions as follows.

(∀/x), (∀/y), (∃/x), (∃/y), etc.

But this seems unnecessary.

Now, we are in a position to formalize

the F is not G

so that we show the two different scopes that are possible.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1xFx/y) ∼ Gy</td>
<td>(1xFx/y) has wide scope</td>
</tr>
<tr>
<td>∼ (1xFx/y) Gy</td>
<td>∼ has wide scope</td>
</tr>
</tbody>
</table>

Before we continue, it is important to observe that the notion of description scope can be generalized to all singular-terms. This gives us scoped-terms, which are defined as follows.

(d5) Where τ is any singular-term, and where ν is any variable, the expression (τ/ν) is a scoped-term.

Grammatically speaking, scoped-terms are not singular-terms, but sentential adverbs, like quantifiers. This is formulated in the following grammatical rule.

(R) if Φ is a formula, and (τ/ν) is a scoped-term, then (τ/ν)Φ is a formula.
In reading formulas containing scoped-terms, the following is the general rule.

\[(\tau/\nu)\Phi\quad \tau \text{ is such that } \Phi\]

\[\text{it is true of } \tau \text{ that } \Phi\]

As it turns out, the general concept of scoped-term affords us grammatical generality, but it does not have much payoff in ordinary first-order logic. Note, however, that it has tremendous payoff in modal logic, where it can be used to clarify the so called *de re/de dicto* distinction.

Grammar is important to logic, but logic is more than grammar. To do logic, we need additionally to formulate inference rules for scoped-terms. As it turns out, the rules are actually quite simple; the in-rule and the out-rule can be combined into a single bi-directional rule – Def \((\tau/\nu)\)

\[\text{Def } (\tau/\nu)\]

\[(\tau/\nu)\Phi\]

\[\therefore \exists v (v = \tau \& \Phi)\]

\(\Phi\) is any formula, \(v\) is any variable, and \(\tau\) is any closed singular-term.

When we apply this new rule to the special case of descriptions, we can deduce the various Russell formulas concerning scope.

(1) \((1xFx/y)Gy\) Pr [wide ‘1xFx’]  
(2) \(\text{SHOW: } \exists y \{\forall x(Fx \leftrightarrow x=y) \& Gy\}\) DD [Russell formula]  
(3) \(\exists y \{y = 1xFx \& Gy\}\) 1, Def \((\tau/\nu)\)  
(4) \(a = 1xFx \& Ga\) 3, \(\exists o\)  
(5) \(\forall x(Fx \leftrightarrow x=a)\) 4a, \(\exists o\)  
(6) \(\forall x(Fx \leftrightarrow x=a) \& Ga\) 4b, 5, SL  
(7) \(\exists y \{\forall x(Fx \leftrightarrow x=y) \& Gy\}\) 6, \(\exists l\)

These two derivations give us the following equivalence

\((1xFx/y)Gy \leftrightarrow \exists y \{\forall x(Fx \leftrightarrow x=y) \& Gy\}\)

By very similar reasoning, we can produce the following equivalences.

\((1xFx/y)\sim Gy \leftrightarrow \exists y \{\forall x(Fx \leftrightarrow x=y) \& \sim Gy\}\)

\(\sim (1xFx/y)Gy \leftrightarrow \sim \exists y \{\forall x(Fx \leftrightarrow x=y) \& Gy\}\)

These correspond to the different scope readings of ‘the F’ in relation to the negation operator. In the first case, ‘the F’ is wide; in the second case, ‘the F’ is narrow.