1. **Number-Words**

By a *number-word* – or *numeral*\(^1\) – we mean a word (or word-like compound\(^2\)) that denotes a number.\(^3\) In English, number-words *appear* to be used as quantifiers, as in the following examples.

- two dogs are barking
- Jay owns three dogs
- there are four dogs in the yard

On the other hand, number-words also figure in the following sorts of constructions.

- the three dogs
- my four dogs
- all five dogs
- no two dogs are exactly alike
- we three kings (of Orient, are bearing gifts, we traverse afar…)

Supposing number-words are quantifiers, these violate the stricture prohibiting double-determiners, so these examples suggest that number-words are not quantifiers.

In order to account for these data, we propose that number-words are fundamentally adjectives, and the first three examples involve in definite-noun phrases, on a par with the following parallel sentences, which we deal with in a separate chapter.

- a dog is barking
- dogs are barking
- Jay owns a dog
- Jay owns dogs

2. **Numerical-Adjectives**

We initially propose to treat numerals as *bare-adjectives* [type C].\(^4\) For example, the poetic

- now we are three

may be syntactically analyzed as follows.

<table>
<thead>
<tr>
<th>now</th>
<th>we [+1]</th>
<th>are</th>
<th>three</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-adverb</td>
<td>DNP [+1]</td>
<td>copula</td>
<td>bare-adjective</td>
</tr>
<tr>
<td>D₁</td>
<td>C→(D₁→S)</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>S→S</td>
<td>D₁→S</td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) The term ‘numeral’ is often reserved for *writing systems* – in particular, for special *logograms* that denote numbers, including Arabic numerals like \(2\), Roman numerals like \(II\), and Hanzi numerals like \(二\).

\(^2\) We are principally interested in *basic* number-words, which is to say the words and word-like compounds that children proudly recite – including, for example:

- one, fifty, two hundred, three thousand sixty-five,
- six million three hundred thousand one hundred twenty five

In this chapter, we take for granted the morphology/semantics of basic number-words, which is a topic unto itself, which is dealt with in the chapter “The Morphology of Number Words”.

\(^3\) What precisely number are (metaphysically speaking) is not our concern here.

\(^4\) Numerals are used in combination with measure-nouns (like ‘acres’ and ‘gallons’), so for these applications we must treat numerals as *modifier-adjectives* [type C→C] that are moreover *not conjunctive*: to be three-gallons is not to be three and gallons! On the other hand, to be three-gallons of water is to be three-gallons and to be water. However, for the special case of count-nouns, we can *pretend* that numerals are bare-adjectives.
With this in mind, we offer the following lexical entries.

\[
\begin{align*}
\text{one} & = \lambda x_0 \text{1}[x] \\
\text{two} & = \lambda x_0 \text{2}[x] \\
\end{align*}
\]

Here the numerical expressions in the semantic-language are understood as follows.

\[
\begin{align*}
1[\alpha] & = \alpha \text{ is one} \\
2[\alpha] & = \alpha \text{ are two} \\
3[\alpha] & = \alpha \text{ are three} \\
\end{align*}
\]

etc.

Note that, in order for these to make sense, we must posit **plural-entities**, in addition to the usual **singular-entities**. The following is an example.

1. **those are three dogs**

\[
\begin{array}{|c|c|c|c|}
\hline
\text{those} & [+1] & \text{are} & \text{three} & \text{dogs} \\
\hline
\delta & \lambda x_0 x_1 & \lambda x_0 3x & \lambda x_0 Dx \\
\delta_1 & \lambda P_0[P_1] & \lambda x_0(3x & \& Dx) \\
\hline
\end{array}
\]

\[
\begin{align*}
\delta & = \lambda x_1(3x & \& Dx) \\
\delta & = \lambda x_1(3x & \& Dx) \\
\end{align*}
\]

Note that \( \delta \) is understood to be a demonstrated plural-entity. Also note that the common noun ‘dogs’ \([D]\) is understood to apply to every plurality of dogs.

3. **Indefinite Article**

Recall that we have so far not offered an account of the indefinite article ‘a’. We now correct that.

\[
\begin{align*}
\text{type}(a) & = C \\
\text{trans}(a) & = \lambda x_0 1x \\
\end{align*}
\]

In other words, ‘a’ means ‘one’. The following is an example.

2. **that is a dog**

\[
\begin{array}{|c|c|c|c|}
\hline
\text{that} & [+1] & \text{is} & \text{a} & \text{dog} \\
\hline
\delta & \lambda x_0 x_1 & \lambda x_0 1x & \lambda x_0 Dx \\
\delta_1 & \lambda P_0[P_1] & \lambda x_0(1x & \& Dx) \\
\hline
\end{array}
\]

\[
\begin{align*}
\delta & = \lambda x_1(1x & \& Dx) \\
\delta & = \lambda x_1(1x & \& Dx) \\
\end{align*}
\]

Of course, in elementary logic, the domain includes **only** singular-entities [no plural-entities, no mass-entities], a result of which is that ‘a’ is semantically redundant.

---

5 This does not seem so implausible when one notices that many languages – e.g., German, French, Spanish, … – use the same word-form – eine, une, una, … [variously inflected!] – to mean both ‘a’ and ‘one’.
4. Numeral-Headed Phrases as Subjects

If numerals are officially adjectives, then we face difficulties analyzing examples in which numeral-headed phrases serve as subjects or objects, such as the following.

3. two dogs are barking

<table>
<thead>
<tr>
<th>two</th>
<th>dogs</th>
<th>[+1]</th>
<th>are barking</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda x_0 2x)</td>
<td>(\lambda x_0 Dx)</td>
<td>(\lambda x_1)</td>
<td></td>
</tr>
<tr>
<td>(\lambda x_0 {2x &amp; Dx})</td>
<td>(\lambda x_1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The problem is that ‘two dogs’ ['three dogs'] is a C, and accordingly cannot enter into a sentence as a subject [object], at least according to the current semantic-scheme.

5. Numerical-Quantifiers

The use of common-noun-phrases as subjects/objects is a well-known problem in semantics, which we discuss in the chapter "Indefinite Noun Phrases".

In this chapter, we sidestep this issue entirely, and instead exclusively consider genuine numerical-quantifiers. In this connection, we begin by noting that

I own three dogs

might be considered ambiguous between

I own at least three dogs

and

I own exactly three dogs

Whereas ‘three’ by itself is an adjective, the latter two phrases ‘exactly three’ and ‘at least three’ are quantifiers. For example, the phrases

the three dogs
my four dogs
all five dogs
no two dogs are exactly alike
we three kings
become infelicitous when we insert ‘exactly’ or ‘at least’, as seen in the following examples.\textsuperscript{6}

- the at least three dogs
- my exactly four dogs
- all at least five dogs
- no exactly two dogs are exactly alike
- we at least three kings


Whereas ‘one’, ‘two hundred’, etc., are best understood as adjectives rather than quantifiers, ‘exactly one’ and ‘at least two hundred’ are best understood as quantifiers. How do we semantically render the phrases ‘exactly’ and ‘at least’?

Earlier, we saw how to analyze various quantifiers – including ‘every’, ‘some’, ‘no’ – in terms of infinitary-operators (a.k.a. junctions). The same approach can be used for numerical-quantifiers.

1. At Least N

First, the notion of ‘at least’ [as attached to number-words\textsuperscript{7}] can be analyzed as follows.

$$\text{[at least]} = \lambda N_0 \lambda P_0 \{ x \mid N_0 x \& P_0 x \}$$

Here the variable ‘\(N\)’ ranges over numerical-adjectives in the semantic-language. The following are example calculations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jay</td>
<td>$\lambda x \lambda y_1 Oyx$</td>
<td>$\lambda N_0 \lambda P_0 { x \mid N_0 x &amp; P_0 x }$</td>
<td>$\lambda x_0, 3x$</td>
</tr>
<tr>
<td></td>
<td>$\lambda P_0 { x \mid 3x &amp; P_0 x }$</td>
<td>$\lambda x_0, 3x$</td>
<td>$D_0$</td>
</tr>
<tr>
<td></td>
<td>$\lor { x \mid 3x &amp; D_0 x }$</td>
<td>$\lambda x_0, 3x$</td>
<td>$\lambda x, x_2$</td>
</tr>
<tr>
<td>$J_1$</td>
<td>$\lambda y_1 Oyx$</td>
<td>$\lor { x \mid 3x &amp; D_0 x }$</td>
<td>$\lor { 3x &amp; D_0 x }$</td>
</tr>
<tr>
<td></td>
<td>$\lor { O_1 x \mid 3x &amp; D_0 x }$</td>
<td>$\lor { 3x &amp; D_0 x }$</td>
<td>$\lor { O_1 x \mid 3x &amp; D_0 x }$</td>
</tr>
<tr>
<td></td>
<td>$\exists x { 3x &amp; D_0 x }$</td>
<td>$\exists x { 3x &amp; D_0 x }$</td>
<td>$\exists x { 3x &amp; D_0 x }$</td>
</tr>
</tbody>
</table>

The final formula says that there is a plurality \(x\) that is 3-big, and is dogs, and is owned by Jay; alternatively, there is a dog-trio that Jay owns. Presumably, to own a dog-trio is to own each individual dog in this trio.

\textsuperscript{6}This is not to say that they are completely incomprehensible, only that they are syntactically ill-formed. The alarm they raise is primarily syntactic, not semantic.

\textsuperscript{7}The expression ‘at least’ is an all-purpose adverb; at least, that’s what I’ve heard.
2. **at least one dog is barking**

<table>
<thead>
<tr>
<th>at-least</th>
<th>one dog</th>
<th>[+] is barking</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda N_0 \lambda P_0 \lor { x \mid N_x \land Px }$</td>
<td>$\lambda x_0 _x$</td>
<td>$D$</td>
</tr>
<tr>
<td>$\lambda P_0 \lor { x \mid I_0 \land Px }$</td>
<td>$\lor x_0 _x$</td>
<td>$\lambda x_x$</td>
</tr>
<tr>
<td>$\lor { x \mid I_x \land Dx }$</td>
<td>$\lambda x_x$</td>
<td>$B_x$</td>
</tr>
<tr>
<td>$\lor { B_x \mid I_x \land Dx }$</td>
<td>$\lor x_x$</td>
<td>$B_x$</td>
</tr>
<tr>
<td>$\exists{ I_0 \land Dx \land B_x }$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice that the output node reads this sentence as semantically-equivalent to ‘some dog is barking’, so long as we presume in the latter that ‘dog’ means dog-individual. The ‘at least one’ locution makes this presumption completely clear.\(^8\)

### 2. Exactly \( N \)

Having dealt with ‘at least’, we next consider ‘exactly’ [as used with number-words\(^9\)]. In order to deal with this, we introduce yet another infinitary-operator, $\boxtimes$, which corresponds to (anadic) **exclusive-or**, which has the following semantic property.\(^{10}\)

$$\exists \{ P\_1, ..., P\_k \} \text{ is true }$$

iff

exactly one of $\{ P\_1, ..., P\_k \}$ is true

We next offer the following analysis of ‘exactly’.

$$[\text{exactly}] = \lambda N\_0 \lambda P\_0 \boxtimes \{ x \mid N\_x \land Px \}$$

For example,

1. $[\text{exactly one dog}] = \boxtimes \{ x \mid I\_x \land Dx \}$
2. $[\text{exactly three dogs}] = \boxtimes \{ x \mid 3\_x \land Dx \}$

etc.

where the right hand sides of (1) and (2) respectively amount to:

$$\boxtimes \{ 1\text{-dog}_1, 1\text{-dog}_2, ..., 1\text{-dog}_k \}$$

$$\boxtimes \{ 3\text{-dog}_1, 3\text{-dog}_2, ..., 3\text{-dog}_k \}$$

where $\{ 1\text{-dog}_1, 1\text{-dog}_2, ..., 1\text{-dog}_k \}$ is the set of all solo-dogs, $\{ 3\text{-dog}_1, 3\text{-dog}_2, ..., 3\text{-dog}_k \}$ is the set of all dog-trios, and $\boxtimes$ is the exclusive-or operator.

---

\(^{8}\) Besides, the prospect of dog-matter barking is very creepy.

\(^{9}\) ‘exactly’ is an adverb with wide application – like, but not exactly like, ‘at least’.

\(^{10}\) It is vital to understand that exclusive-or is best understood as an anadic operator, not a dyadic operator. Whereas this distinction makes little or no difference for ‘and’ and inclusive-or, it is vital for exclusive-or. Treating exclusive-or as a dyadic operator produces undesirable (even ridiculous) results, as seen in the following identities.

\[ (T \text{xor} T) \text{xor} T = F \text{xor} T = T \]

versus

\[ \text{xor} \{ T, T, T \} = F \]
The following is an example calculation.

3. **Jay owns exactly three dogs**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>λyornaλx,yOxy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>λy[3y &amp; Dy]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>λy[3y &amp; Oy]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>λx,x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>λN0λP0{y</td>
<td>Ny &amp; Py}</td>
<td>3x</td>
<td>D0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λP0{y</td>
<td>3y &amp; Py}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lines ①-③ are constructed by the usual junction-composition rule.\(^{11}\)

\[ \text{\textbf{\textit{\textbf{\mathit{\mathcal{J}}}-Composition}}} \]

- $\alpha$, $\beta$, $\gamma$ are any expressions;
- $\mathcal{J}\{ \beta | \Phi \}$ $\Phi$ is any formula;
- $\alpha ; \beta \vdash \gamma$ any sub-derivation of $\gamma$ from $\{\alpha, \beta\}$
- $\mathcal{J}\{ \gamma | \Phi \}$ $\mathcal{J}$ admits $\alpha$

Line ③ amounts to the following.

$\mathcal{O}[J, \text{dog-trio}_1] \mathcal{\&} \ldots \mathcal{O}[J, \text{dog-trio}_k]$

Given the nature of exclusive-or, this amounts to saying that there is exactly one dog-trio that Jay owns. Now, this is also precisely what ⑤ says, but using the familiar logical notation ‘∃!’ (“there is exactly one…”). The latter move is underwritten by the following type-logical principle.

\[ \text{\textbf{\textit{\textbf{\mathcal{S}}}-Simplification}} \]

- $\Phi, \Psi$ are formulas
- $\mathcal{S}\{\Psi | \Phi\}$ $\nu$ are the variables c-free in $\Phi$
- $\exists!\nu\{\Phi \& \Psi\}$ $\exists!\nu\Omega =_{\text{def}} \exists w \forall \nu\{\Omega \leftrightarrow \nu = w\}$

In the latter definition, $\nu$ is any sequence of variables, and $w$ is any sequence of variables that is completely distinct from $\nu$ and not free in $\Omega$; identity then corresponds to component-wise identity.

Finally, line ⑤ is obtained from ⑤ in accordance with notation and principles explained in the following section.

\(^{11}\) We tentatively postulate that $\mathcal{S}$ admits everything.
7. Second-Order Numerical-Predicates

Thus far, we use numerals in the semantic-language as bare-adjectives \([D_0 \rightarrow S]\), informally characterized as follows.

\[1[\alpha] = \alpha \text{ is one}\]
\[2[\alpha] = \alpha \text{ are two}\]

etc.

We can also use numerals in the semantic-language as second-order predicates \([(D \rightarrow S) \rightarrow S]\), characterized as follows.\(^\text{12}\)

\[1[P] = \exists x \forall y (Py \leftrightarrow y=x)\]
\[2[P] = \exists x_1 \exists x_2 \{ x_1 \neq x_2 \land \forall y (Py \leftrightarrow y=x_1 \lor y=x_2) \}\]
\[3[P] = \exists x_1 \exists x_2 \exists x_3 \{ x_1 \neq x_2 \land x_1 \neq x_3 \land x_2 \neq x_3 \land \forall y (Py \leftrightarrow y=x_1 \lor y=x_2 \lor y=x_3) \}\]

etc.

We also introduce the following natural shorthand,

\[N \nu \Phi =_d N[\lambda \nu \Phi]\]

where \(\nu\) is a variable, \(\Phi\) is a formula, and \(N\) is a second-order numerical-predicate (as above).\(^\text{13}\)

8. Scope-Ambiguity

When quantifier-phrases are combined, we often have scope-ambiguities, and numerical-quantifiers are no exception, as illustrated in the following example.

exactly-two men respect exactly-two women

Indeed, this has three readings, since it can be used to answer three quite different questions.\(^\text{14}\)

1. how many men respect exactly-two women?
2. how many women are respected by exactly-two men?
3. how many men respect how many women?

\(\text{12}\) Although one might prefer to use two different sets of numerals in the semantic-language, one for adjectives, the other for quantifiers, we choose to use one set ambiguously. The context will indicate the intent.

\(\text{13}\) These definitions only make sense when talking about count-items. Of course, from the viewpoint of the semantic-language, all items are count-items, including those items we use to model mass-items.

\(\text{14}\) We presume a simple-minded account of respect according to which it stands between individuals and not groups of individuals. Otherwise, there are further (collective) readings as well. Also, ‘respect’ is stative. When eventive verbs and nouns are involved, counting is even more complicated. For example, what does the following mean?

Last year, the world’s busiest airport [ATL (Atlanta)] served 90,039,280 passengers, and the world’s second busiest airport [ORD (Chicago)] served 69,353,654 passengers. What exactly are passengers, and how do we count them? Here, a passenger is more like a "passenging".
The first two are computed, using **serial-composition**, respectively as follows.\(^{15}\)

1. exactly two men respect exactly two women [‘exactly two men’ wide]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda y \lambda x Rxy )</td>
<td>( \exists { \lambda x Rxy</td>
<td>2Wy } )</td>
</tr>
<tr>
<td>( \exists ! { \lambda x Rxy</td>
<td>2Wy } )</td>
<td></td>
</tr>
</tbody>
</table>

Here, in deriving \( \exists ! \) from \( \exists \), the first junction absorbs the second junction. By contrast, in the following, the second junction absorbs the first junction.

2. exactly two men respect exactly two women [‘exactly two women’ wide]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda y \lambda x Rxy )</td>
<td>( \exists { \lambda x Rxy</td>
<td>2Wy } )</td>
</tr>
<tr>
<td>( \exists ! { \lambda x Rxy</td>
<td>2Wy } )</td>
<td></td>
</tr>
</tbody>
</table>

The third reading is more subtle and involves parallel-composition.

3. exactly two men respect exactly two women [equal/parallel scope]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda y \lambda x Rxy )</td>
<td>( \exists { \lambda x Rxy</td>
<td>2Wy } )</td>
</tr>
<tr>
<td>( \exists ! { \lambda x Rxy</td>
<td>2Wy } )</td>
<td></td>
</tr>
</tbody>
</table>

\( \exists ! \) says that there is exactly one pair \( \langle x, y \rangle \) of *pluralities* such that \( x \) are two-men and \( y \) are two-women and \( x \) respect \( y \).\(^{16}\) Also note that, in deriving \( \exists ! \) from \( \exists \), the junctions are combined by parallel-composition, given as follows, where \( \emptyset \) is any junction.

### Parallel-Composition

| \( \emptyset \{ \alpha | \Phi \} \) | \( \alpha, \beta, \gamma \) are any expressions; |
| \( \emptyset \{ \beta | \Psi \} \) | \( \Phi, \Psi \) are any formulas; |
| \( \alpha ; \beta \vdash \gamma \) | any sub-derivation of \( \gamma \) from \( \{ \alpha, \beta \} \) |
| \( \emptyset \{ \gamma | \Phi \& \Psi \} \) | |

\(^{15}\) Henceforth, we adopt the following natural short-cut, where \( A \) is a bare-adjective and \( B \) is a noun.

\[
AB\alpha =_\emptyset A\alpha \& B\alpha
\]

\(^{16}\) Presumably *respect* is distributive – \( x \) respect \( y \) iff every member of \( x \) respects every member of \( y \).