Morphology of Numerals

Gary Hardegree
Department of Philosophy
University of Massachusetts
Amherst, MA 01003

1. Morphology 2
2. Numbers and Numerals 2
3. A Simple Numerical Grammar – NG1 3
4. A Brief Aside on Rewrite Rules 4
5. An Example of a Tree in NG1 6
6. Semantics for NG1 7
7. The Semantics of NG1 is not Local 8
8. A Variation of NG1 – NG2 9
   1. Syntax 9
   2. Semantics of NG2 10
9. A Syntactic Problem with NG1 and NG2 10
10. A More Adequate Numerical Grammar – NG3 11
   1. Two-Digit Numerals 11
   2. Three-Digit Numerals 11
   3. Numerals Written with one Comma 12
   4. Numerals Written with two Commas 13
11. The Official Grammar – NG3 14
   1. Syntax 14
   2. Examples of Phrase-Structures 15
   3. Phonology 16
   4. Semantics of NG3 18
12. A Brief Note On European-Style Numeral Morphology 19
13. Categorial (Fregean) Reconstruction of NG3 – NG3f 19
14. English Number-Words – ENG1 21
   1. Syntax 21
   2. Semantics 21
   3. Phonology 21
   4. Example 22
15. A Minor Shortcoming of ENG1 23

1. Morphology

Morphology is the branch of grammar that pertains to the structure of words.[1] The smallest units of morphology are called 'morphemes'. Probably the simplest example of a morphological analysis is the analysis of a plural noun into a noun stem, and a plural suffix; for example, 'cats' = 'cat' + 's'.

In this chapter, we discuss the morphology of *number-words*, as a way of testing and illustrating the basic ideas of grammar – including syntax, semantics, and phonology. By 'number words' we mean to include both phonograms (like 'one hundred') and the logograms (like '100' and 'CXX'). We will call both kinds of words 'numerals'.

2. Numbers and Numerals

The distinction and connection between numbers and numerals is quite simple – numerals are syntactic objects that denote numbers. The following illustrate the connection.

the numeral '0' denotes the number 0
the numeral '1' denotes the number 1
the numeral '2' denotes the number 2
etc.

Whereas numerals are symbols, numbers are quantities.[2] Numbers are the possible answers to questions of the form "how many…".[3]

Basically, numerals come in two syntactic forms – quantifiers and proper nouns. For example, in the sentence

there are six students in this class

the word 'six' serves as a quantifier. A quantifier, in turn, may be understood as a second-order adjective/predicate.

Many adjectives yield corresponding proper nouns. For example, the word 'blue' is both an adjective and a proper noun, as illustrated in the following sentences.

my favorite shirt is blue
my favorite color is blue

Elementary logic presents a similar example; specifically, the terms 'true' and 'false' are used both as adjectives and as proper nouns (although the latter are admittedly contrived). For example:

the sentence 'snow is white' is true
the truth-value of 'snow is white' is true

The same holds of second-order adjectives (quantifiers) like 'one', 'two', etc. The following illustrate.

---

[1] The word also has a meaning in Biology, which pertains to "… the form and structure of organisms without consideration of function." (*American Heritage Dictionary*).

[2] We concentrate on what are standardly called *cardinal numbers*. There are also *ordinal numbers*, which we ignore in this chapter. There are also numerical labels (as in 'room number 163'), which we also ignore.

[3] We concentrate on the so-called *natural numbers* – 0, 1, 2, … When we consider "how much" questions, we are forced to consider a wider class of numbers.

there are six students
the number of students is six

The relation between the adjectives and their corresponding proper nouns is fairly straightforward, being illustrated by the following principles.

a sentence is true	↔	its truth-value is true
T[s]	↔	$\upsilon(s) = t$
a shirt is blue	↔	its color is blue
B[s]	↔	$c(s) = b$
there are two students	↔	the number of students is two
2[S]	↔	$\#(S) = 2$

It is natural to suppose that, although adjectives and proper nouns are quite different syntactically, in each case above at least, the respective adjective and proper noun nevertheless convey the same concept.

The history of numeration systems is long and tortuous, but over the past few hundred years the world has largely adopted the decimal Hindu-Arabic numeral system, which is so ingrained in our culture that we hardly notice it.[4] It is nevertheless a hard-won linguistic achievement.

In this chapter, we examine the grammar (i.e., morphology) of the standard numeration system, first the Hindu-Arabic logogramic system, and then the corresponding English phonogramic system. We also briefly examine the Roman numeral system. We begin with a very simple and natural account of the syntax and semantics of Hindu-Arabic numerals. We then observe that this account has both syntactic shortcomings and semantic shortcomings, which lead us to propose a more sophisticated grammar.

3. A Simple Numerical Grammar – NG1

The standard numeration system is based on ten primitive symbols – the *atomic numerals*, or "digits" – which are then *concatenated* to form *molecular numerals*. As an initial simple account of the syntax, we propose the following.

(1) ‘0’ is an atomic numeral;
‘1’ is an atomic numeral;
…
‘9’ is an atomic numeral;
nothing else is an atomic numeral.

(2) every atomic numeral is a numeral;
if n is a numeral, and a is an atomic numeral, then n+a is a numeral;
nothing else is a numeral.

[4] This numeration system, which traces to India some time in the 6th Century AD, was passed to the West in the 8th Century AD, *via* Baghdad, and finally reached Europe in 1202 AD, when Leonardo of Pisa (a.k.a. Fibonacci) published his *Book of the Abacus*. Fibonacci grew up in Northern Africa, where his father was a diplomat, and where he learned of the new numeration scheme.

Here, the symbol '+' refers to string addition; two strings are added by placing the second one immediately after the first one. For example, '12'+'34' = '1234'.

These are composition rules in the style familiar in logic. The corresponding decomposition ("rewrite") rules, in the style of generative grammar, are given as follows, where 'N' refers to the category of numerals, and 'A' refers to the category of atomic numerals.

(1)	N	⇒	N + A
(2)	N	⇒	A
(3)	A	⇒	'0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

4. A Brief Aside on Rewrite Rules

As we propose to notate them, rewrite rules employ three kinds of expressions.

(1)	special symbols		⇒	+	|
(2)	category names	e.g.,	N	A	
(3)	lexical names	e.g.,	'0'	'1'	etc.

(1a) The arrow-symbol (⇒) is officially read "may be rewritten as", but this may not be all that helpful in the absence of examples, so it is best to hang on a moment while the other notation is explained.

(1b) The plus-symbol (+) is *general syntactic composition*. In the simplest cases at least, syntactic composition is simply string-addition.

(1c) The vertical bar (|) offers a useful abbreviation that saves us from writing a number of very similar clauses.[5] For example, clause (3) above may be understood as ten different clauses:

(3.0) A ⇒ '0'

…

(3.9) A ⇒ '9'

Alternatively, we may understand the vertical bar as a kind of disjunction, so that (3) says in effect that an atomic numeral may be rewritten as *any one* of {'0', '1', …, '9'}.

(2) Category names are largely arbitrary, but they usually derive from extant grammatical category names (e.g., 'adjective' or 'verb').

(3) Lexical names, of course, denote the items in the lexicon, which are those expressions which do not admit rewrites. All syntactic trees terminate at lexical items.

Next, we note that rewrite rules generally have the following basic form.[6]

$$K_1 + \ldots + K_m \quad \Rightarrow \quad K_{m+1} + \ldots + K_{m+n}$$

Here, the various K-objects are categories *and* lexical items.

[5] In principle, we can also put together dissimilar clauses. For example, (1) and (2) can be combined to form:

(1|2) N ⇒ N+A | A

But this is not particularly helpful.

[6] In this connection, we treat the vertical bar as simply an abbreviation technique.

We will assume a number of simplifications of this scheme. First, we will concentrate on "context-free" rules, according to which how a category rewrites does not depend upon what other objects are in its vicinity. This yields the following basic form.

$$K_1 \quad \Rightarrow \quad K_2 + \ldots + K_m$$

We will also presume the usual ideas about lexical items; in particular, there are no rewrites of lexical items.

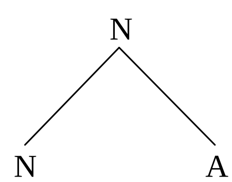
Next, the presence or absence of a plus on the right side divides rewrite rules into two simple categories.

(1)	branching [or, decomposition] rules	one or more pluses
(2)	non-branching [or, subsumption] rules	no pluses

An example of the first type is:

(1) N $\Rightarrow$ N + A

This is a "branching" rule because it authorizes the following branching tree diagram.

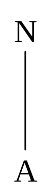


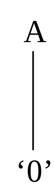
(1) is also said to be a "decomposition" rule because it says that a numeral may be decomposed into two numerals.

The following are examples of the second type.

(2) N $\Rightarrow$ A
(3.0) A $\Rightarrow$ ‘0’

These are "non-branching" rules because they authorize respectively the following non-branching tree diagrams.

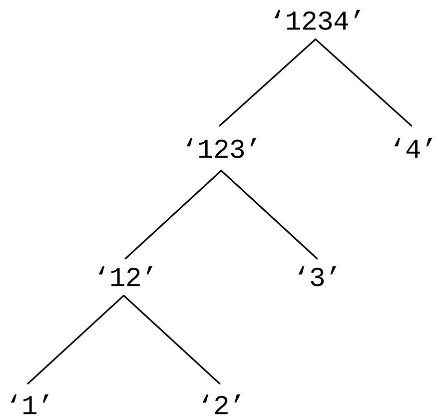


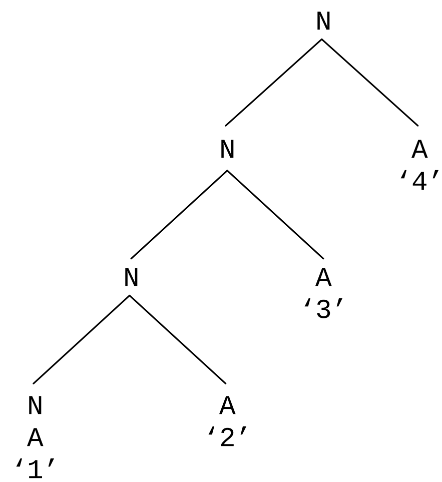


These are also said to be "subsumption" rules because each one says in effect that one category *subsumes* another category. Mathematically speaking, there are two forms of subsumption; on the one hand, A is a *subset* of N; on the other hand, ‘0’ is an *element* of A.

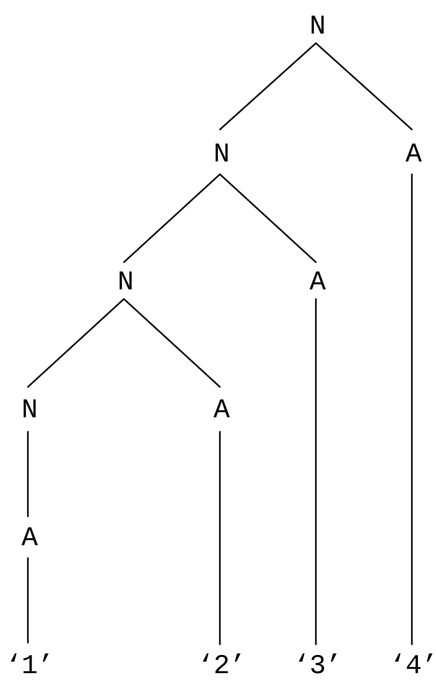
5. An Example of a Tree in NG1

The following is an example of a syntactic-tree – both in the simple mereological form, and in the phrase-marked form.





Note that, in order to save space, we omit lines for subsumption rules, and instead simply write subsumed items directly under the subsuming items. The following is how the tree looks fully expanded, with the lexical items all at the bottom.



6. Semantics for NG1

In principle, in doing a semantics for a generative grammar, we must construct

(1) an entry for each lexical item.
(2) a semantic rule for each rewrite rule.

However, certain rewrite rules can be done *once and for all* using the following *general semantic principle*.

> Subsumption (No Branching) Principle
>
> if α rewrites as β, with *no branching* [i.e., α subsumes β],
> then α inherits the semantic value of β; i.e., $[\![\alpha]\!] = [\![\beta]\!]$

Here, $[\![\alpha]\!]$ is the semantic value of α, which in this case is the denotation of α. In particular, granted the Subsumption Principle, we can skip all non-branching rewrite rules.

Thus, in providing a semantics for NG1, we must provide ten lexical entries, and we must provide a semantic rule corresponding to the rule [N⇒N+A]. These are given as follows.

(1)	$[\![$‘0’$]\!]$	=	0
	$[\![$‘1’$]\!]$	=	1
	…		
	$[\![$‘9’$]\!]$	=	9
(2)	$[\![\mathrm{N} + \mathrm{A}]\!]$	=	$[\![\mathrm{N}]\!] \times 10 \;.\!\!+ [\![\mathrm{A}]\!]$

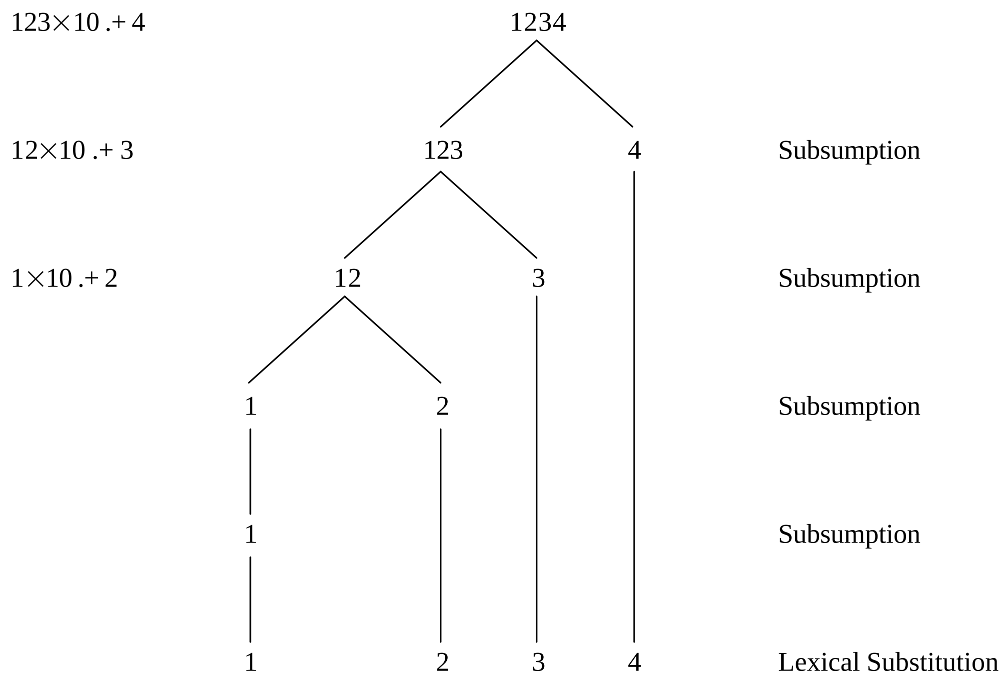
Item (1) is straightforward – the denotation of the numeral ‘0’ is the number 0, the denotation of the numeral ‘1’ is the number 1, etc. Item (2) is not so straightforward, but is a tidy abbreviation of the following.

for any numeral n, and any atomic numeral a, $[\![n{+}a]\!] = [\![n]\!]{\times}10 \;.\!\!+ [\![a]\!]$

For example:

$[\![$‘12’$]\!]$ = $[\![$‘1’ + ‘2’$]\!]$
= $[\![$‘1’$]\!]$ × 10 .+ $[\![$‘2’$]\!]$
= 1 × 10 .+ 2
= 12

The following is an example of a semantic-tree.

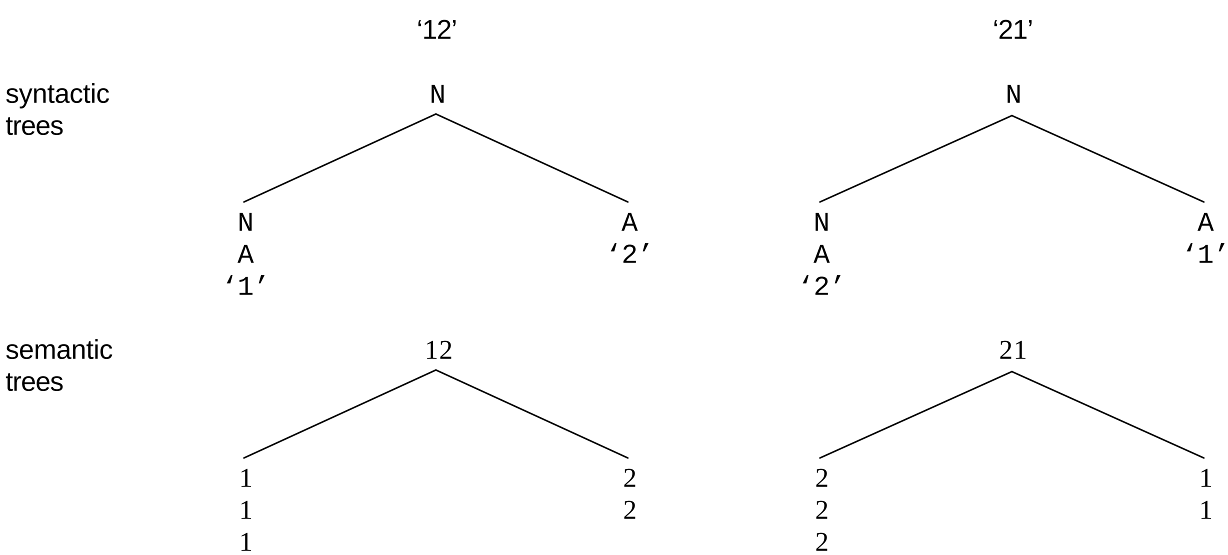


Notice carefully that, whereas the earlier syntactic-tree consists of numerals, the latter semantic-tree consists of numbers.

7. The Semantics of NG1 is not Local

First, we note that NG1 is not categorial, since syntactic-composition is not functor-application. By the same token, NG1 does not satisfy Frege's Thesis, since semantic-composition is not function-application. For example, in the composition of 34 from 3 and 4, neither 3 nor 4 serves as a function that is applied to the other. Rather, semantic-composition is implemented via an *ad hoc* intermediate function that acts on the two inputs.

Maybe we can live with this; after all, the ultimate *desideratum* is compositionality (preferably, locality). Frege's technique is merely a tidy way of achieving locality. The question then is whether NG1 is in local. By way of answering this question, we consider the following four trees.



Recall that the principle of locality requires that the semantic-value of a compound phrase is a function of the semantic-value of its immediate constituents. Notice that, according to NG1, the immediate constituents of ‘12’ and ‘21’ are precisely the same – being ‘1’ and ‘2’ – and yet they denote different numbers [since 12≠21]. The semantic value of a compound is a function of the semantic-values of the immediate constituents plus their left-right order, but *locality* makes no mention of left-right order.

8. A Variation of NG1 – NG2

In the present section, we consider a variation on NG1 – called NG2 – which offers a grammar of numerals that is categorial, Fregean, and local (and hence compositional).

1. Syntax

First of all, NG2 is categorial, so the decomposition rules all resolve to categorial-decomposition, which has the following form.

(c) $K_2 \Rightarrow K_1 \triangleleft (K_1 \rightarrow K_2)$
$K_2 \Rightarrow (K_1 \rightarrow K_2) \triangleright K_1$

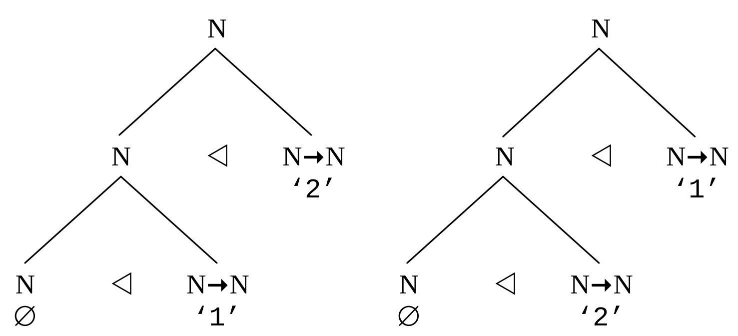
Here the symbol ‘$\triangleright$’ and its converse ‘$\triangleleft$’ refer to functor-application.

In NG2, each atomic numeral is treated as a functor of category N→N; in order to get the ball rolling, it also posits a null-numeral ∅. The rewrite rules for NG2 can then be written as follows.

(1)	N	$\Rightarrow$	N $\triangleleft$ A	[= N $\triangleleft$ (N→N)]
(2)	N	$\Rightarrow$	∅	
(3)	A	$\Rightarrow$	‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’	

Notice in particular that clause (1) says that a numeral results whenever an atomic numeral is applied (from the right) to a numeral.

By way of illustration, we analyze ‘12’ and ‘21’ in accordance with NG2.

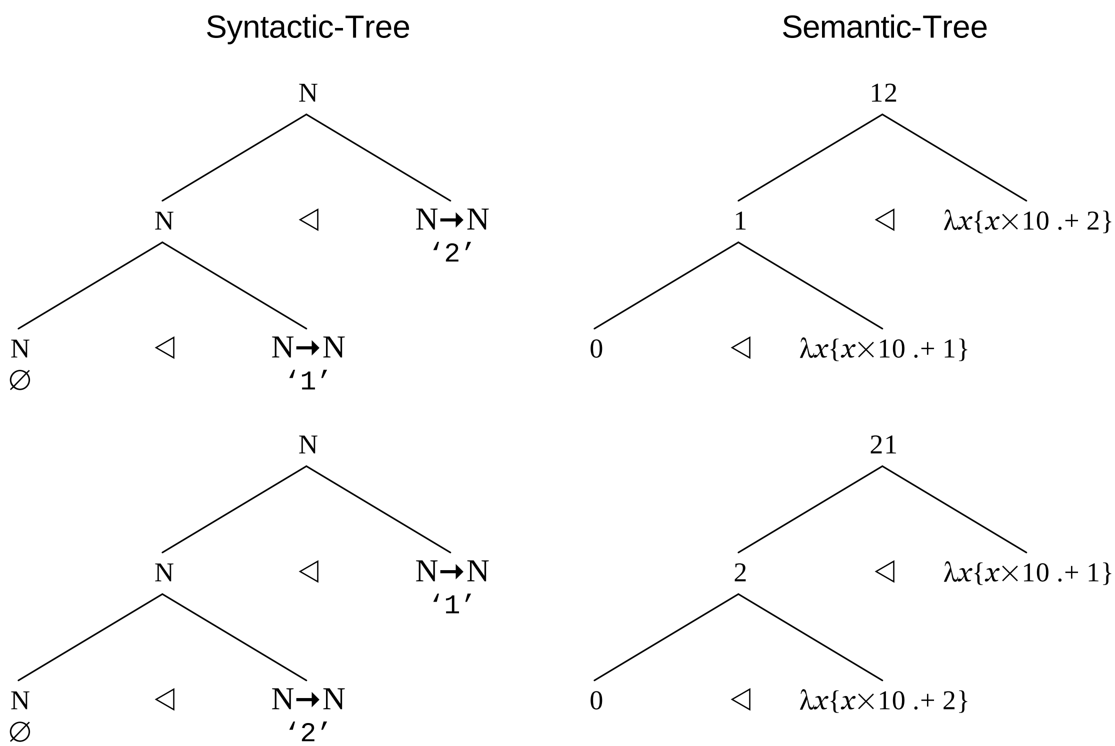


2. Semantics of NG2

Since NG2 is categorial, its semantics may be specified simply by giving the semantic values of all the atomic constituents, which are as follows.

(1)	$[\![\varnothing]\!]$	=	0
(2)	$[\![\text{‘0’}]\!]$	=	$\lambda x\{x\times 10 \,.\!+ 0\}$
	$[\![\text{‘1’}]\!]$	=	$\lambda x\{x\times 10 \,.\!+ 1\}$
	…		
	$[\![\text{‘9’}]\!]$	=	$\lambda x\{x\times 10 \,.\!+ 9\}$

Examples



Notice, in particular, that 12 and 21 do not have the same immediate constituents; whereas 12 has 1 and $\lambda x(x\times 10.\!+2)$ as constituents, 21 has 2 and $\lambda x(x\times 10.\!+1)$ as constituents.

9. A Syntactic Problem with NG1 and NG2

NG2 overcomes the semantic problem we discovered in NG1, but it still has syntactic shortcomings. Specifically, NG1 proposes intermediate constituents that don’t seem to be valid, and there seem to be valid constituents that NG1 does not propose. For example, it seems that the phrase ‘123’ has the following constituents – based on admissible pronunciations in English.

one, hundred, one-hundred, twenty, three, twenty-three,

On the other hand, according to NG1 and NG2, the constituents of ‘123’ are:

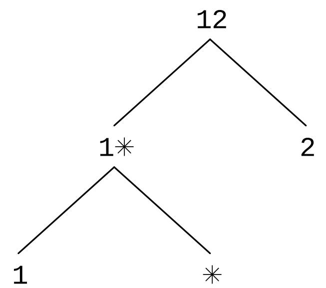
1, 2, 3, 12

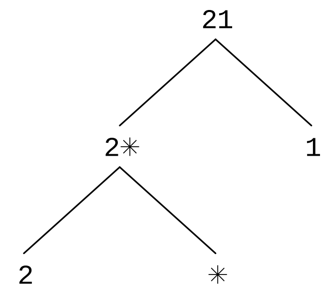
10. A More Adequate Numerical Grammar – NG3

We next consider a numerical grammar that is intended to overcome the syntactic shortcomings of NG2. We develop the grammar piecemeal; first, we give example grammatical analyses; then we propose the official grammar.

1. Two-Digit Numerals

The first problem we must solve is distinguishing the constituents of ‘12’ and ‘21’. For this we propose to interpose a hidden expression between the left digit and the right digit. Using ‘✳’ for this special element, we propose the following two trees, respectively.



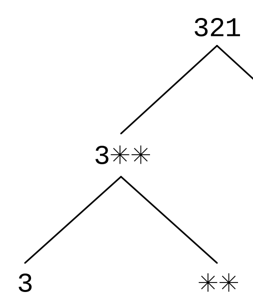


Note: for the time being, we concentrate on syntax, so we drop single quotes. Please keep in mind that we are doing syntax, so the elements under consideration are numerals, not numbers. Notice also that ‘21’ and ‘12’ have different immediate constituents, although they have the same atomic constituents. This is important in achieving locality.

Next, we propose to treat ‘✳’ as an *inflectional* element.[7] For example, the phrase ‘21’ contains an inflected form of ‘2’ – namely ‘2✳’ – which is pronounced “twenty”.

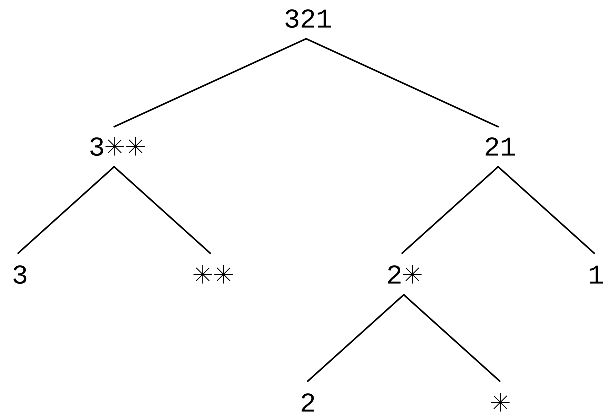
2. Three-Digit Numerals

Next, let’s consider three-digit numerals – for example, ‘321’. Continuing to pursue our inflectional approach, we propose to interpose a further hidden inflectional element between ‘3’ and ‘21’, similar to the one between ‘2’ and ‘1’ in ‘21’? In particular, we propose a further inflectional element, which we notate by ‘✳✳’. Then the tree looks thus.



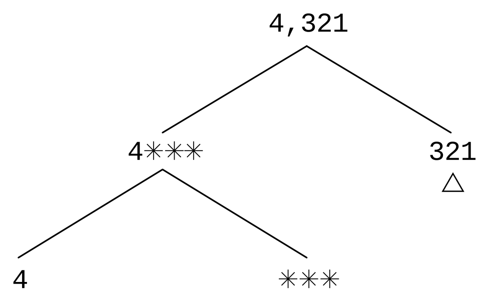
As is customary in grammar, the triangle ‘△’ indicates that there are further constituents in the tree, but ones that we choose to ignore at the moment. If we do choose to expand the sub-tree below ‘21’, we obtain the following.

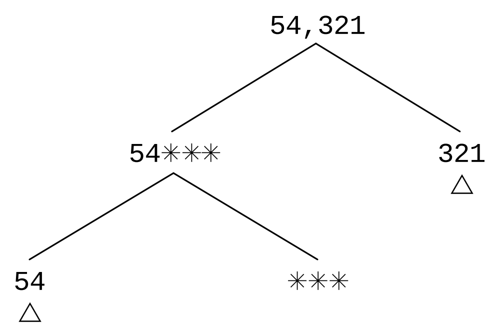
[7] Linguists use the word 'inflection' both in phonology and morphology. In morphology, 'inflection' refers to numerous methods of word "bending", including in particular verb and noun adjustments pertaining to person, number, gender, tense, aspect, and case. For example, the difference between ‘I’ and ‘me’ is an inflectional difference.

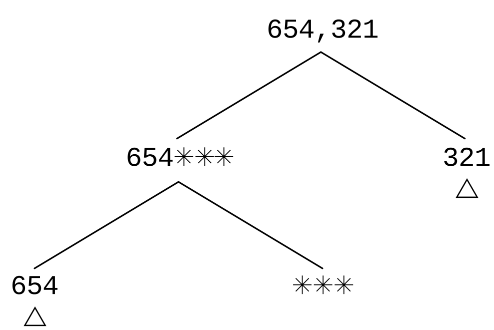


3. Numerals Written with one Comma

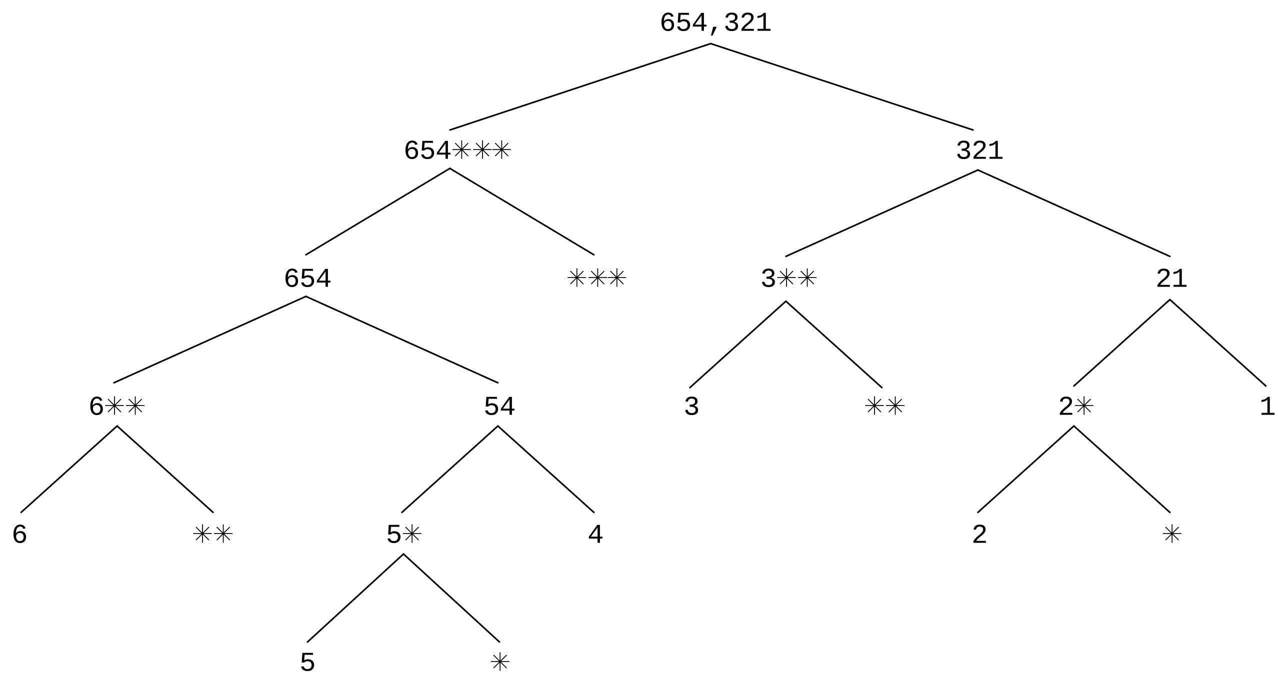
Next, let us consider numerals that are usually written with a single comma. In this connection, we propose a third inflectional element notated by '✳✳✳'. The following are typical examples.





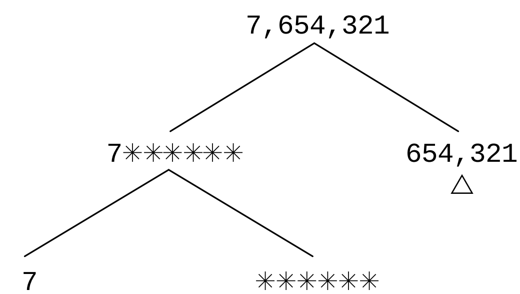


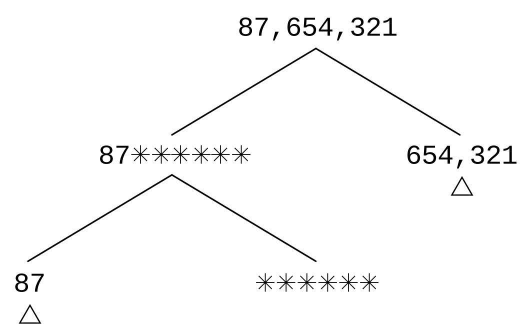
As before, the triangle indicates that there are further constituents we choose temporarily to ignore. In the last tree, if we choose to do a full expansion, we obtain the following.

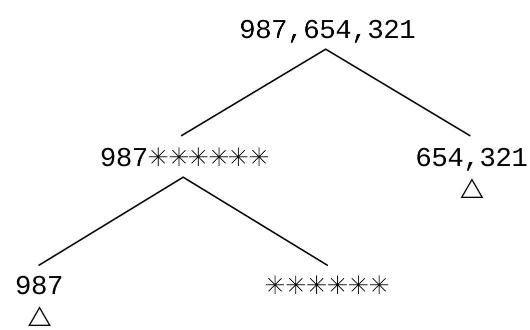


4. Numerals Written with two Commas

The next step is to consider numerals that are usually written using two commas. In this connection, we propose yet another inflectional element, which we notate by ‘✳✳✳✳✳✳’. The following are typical examples.







11. The Official Grammar – NG3

Having done numerous examples, we now officially formulate the rules of NG3.

1. Syntax

The syntax of NG3 is founded on dividing numerals into broad categories according to how many digits they have, as follows.

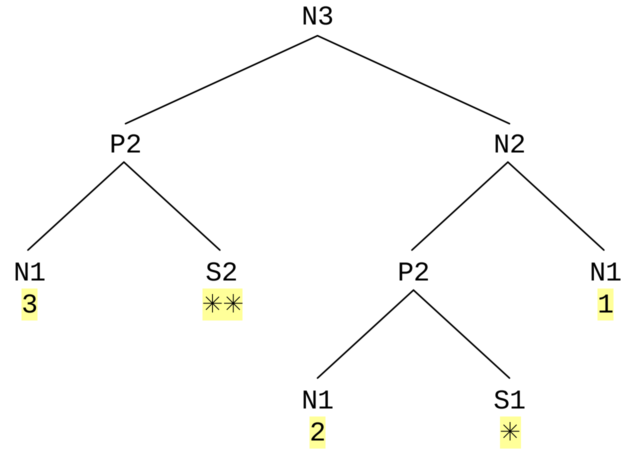
N0	:	‘0’ alone
N1	:	1-digit numerals (other than ‘0’)
N2	:	2-digit numerals
N3	:	3-digit numerals
N6	:	4,5,6 -digit numerals
N9	:	7,8,9 -digit numerals
etc.		

The rewrite rules are given as follows. Notice the category ∅, which is "null".

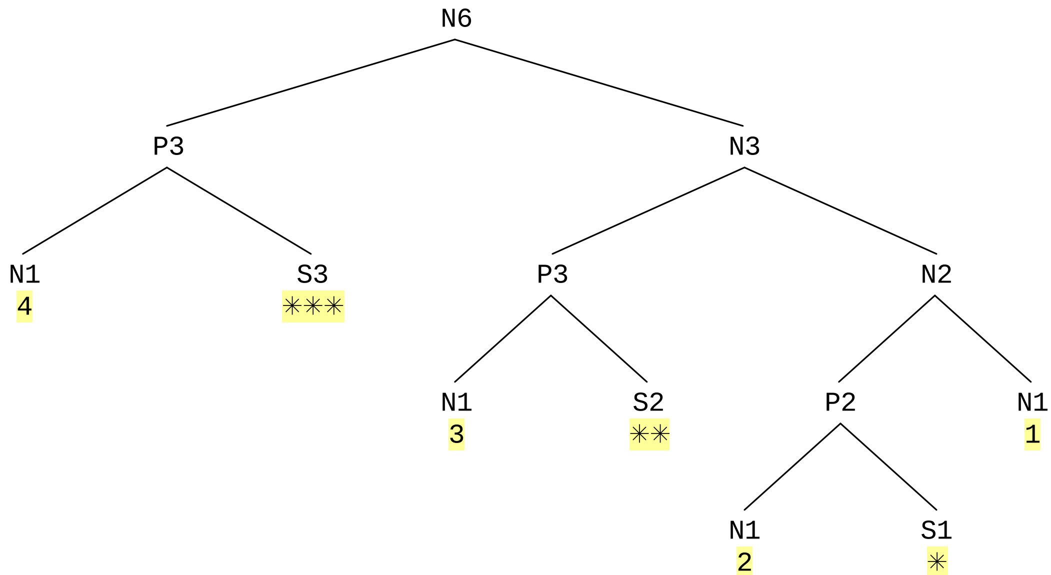
(1)	N	⇒	N0 | N1 | N2 | N3 | N6 | N9 | N12 | …
(2)	N0	⇒	‘0’
	N1	⇒	‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’
	N2	⇒	P1 + { ∅ | N1 }
	N3	⇒	P2 + { ∅ | N1 | N2 }
	N6	⇒	P3 + { ∅ | N1 | N2 | N3 }
	N9	⇒	P6 + { ∅ | N1 | N2 | N3 | N6 }
	N12	⇒	P9 + { ∅ | N1 | N2 | N3 | N6 | N9 }
	etc.		
(3)	P1	⇒	N1 + S1
	P2	⇒	N1 + S2
	P3	⇒	{ N1 | N2 | N3 } + S3
	P6	⇒	{ N1 | N2 | N3 } + S6
	P9	⇒	{ N1 | N2 | N3 } + S9
	etc.		
(4)	S1	⇒	‘∗’
	S2	⇒	‘∗∗’
	S3	⇒	‘∗∗∗’
	S6	⇒	‘∗∗∗∗∗∗’
	etc.		

2. Examples of Phrase-Structures

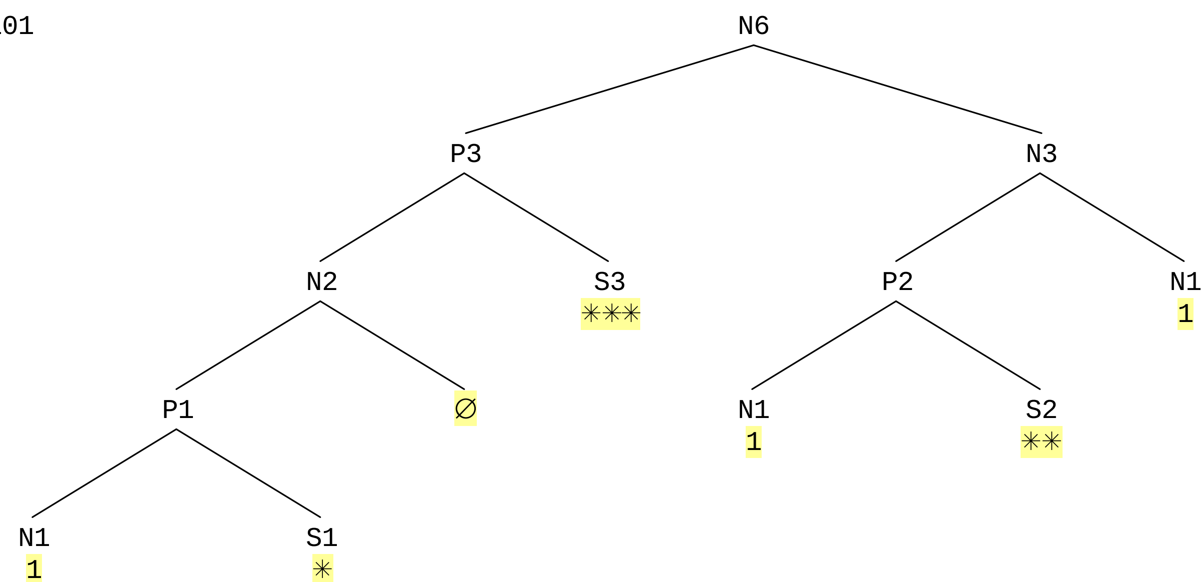
321



4321



10,101



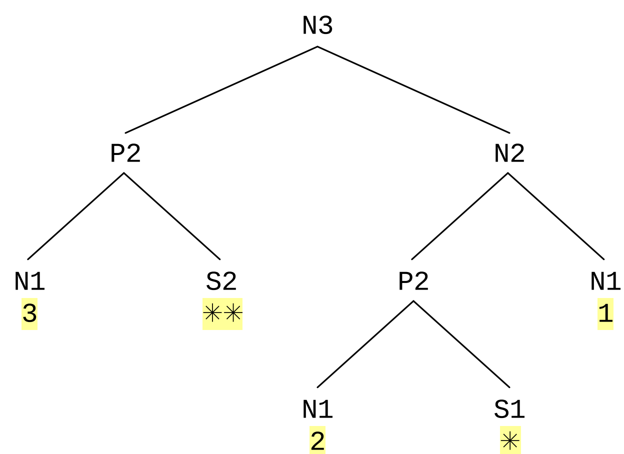
3. Phonology

The phonetic module of our grammar has a fairly simple underlying form, although there are numerous idiomatic transformations. The basic idea is that we read the terminal nodes of the syntactic tree left-to-right applying the following rules.

(1)	∅		unpronounced	
	‘0’	⇒	"zero"	
	‘1’	⇒	"one"	
	etc.			
(2)	✳	⇒	"ty"	[pronounced like ‘tee’]
	✳✳	⇒	"hundred"	
	✳✳✳	⇒	"thousand"	
	✳6✳	⇒	"million"	
	✳9✳	⇒	"billion"	
	etc.			
(3)	"one-ty ∅"	⇒	"ten"	
	"one-ty one"	⇒	"eleven"	
	"one-ty two"	⇒	"twelve"	
	"one-ty three"	⇒	"thirteen"	
	etc.			
(4)	"two-ty"	⇒	"twenty"	
	"three-ty"	⇒	"thirty"	
	etc.			

The following is a simple example of a phonetic transcription.

321



The terminal nodes (i.e., nodes without daughters) are the following

3	✳✳	2	✳	1

Reading these left-to-right in accordance with the first-stage rules yields the following.

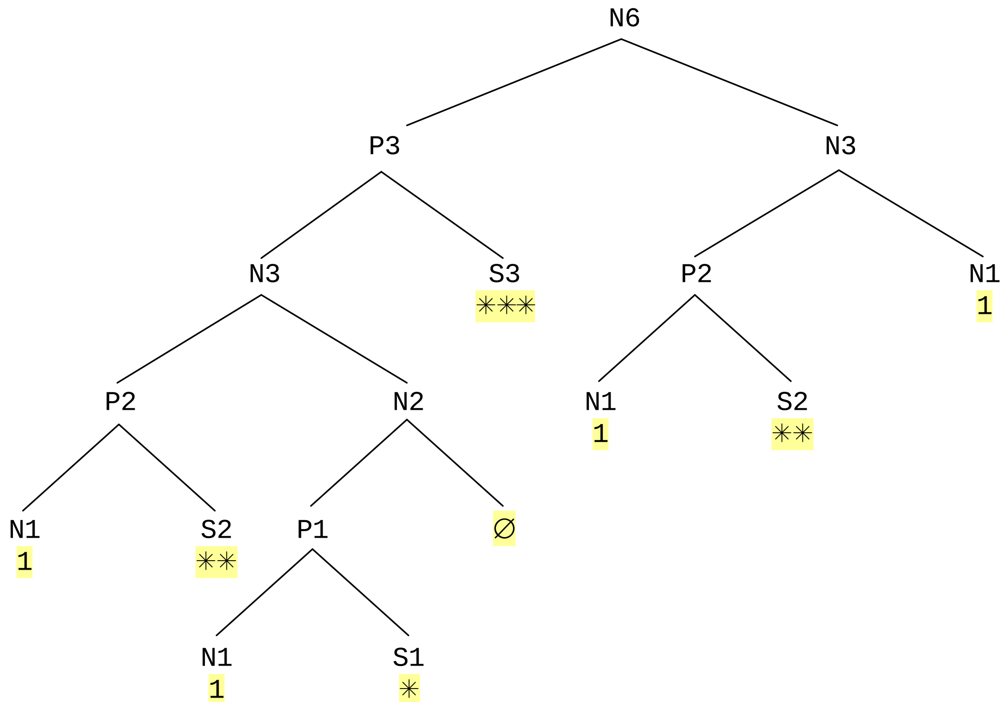
three	hundred	two	ty	one

The bizarre phonetic item "two ty" is in turn transformed into the more familiar phonetic item "twenty", which yields the following.

three	hundred	twenty	one

The following is a somewhat more complicated example.

110,101



In this tree the terminal nodes are:

1	**	1	*	∅	***	1	**	1

Reading these left-to-right in accordance with the first-stage phonetic transcription, we obtain:

one	hundred	one	ty		thousand	one	hundred	one

Further transformations produce the following.

one	hundred	ten		thousand	one	hundred	one

4. Semantics of NG3

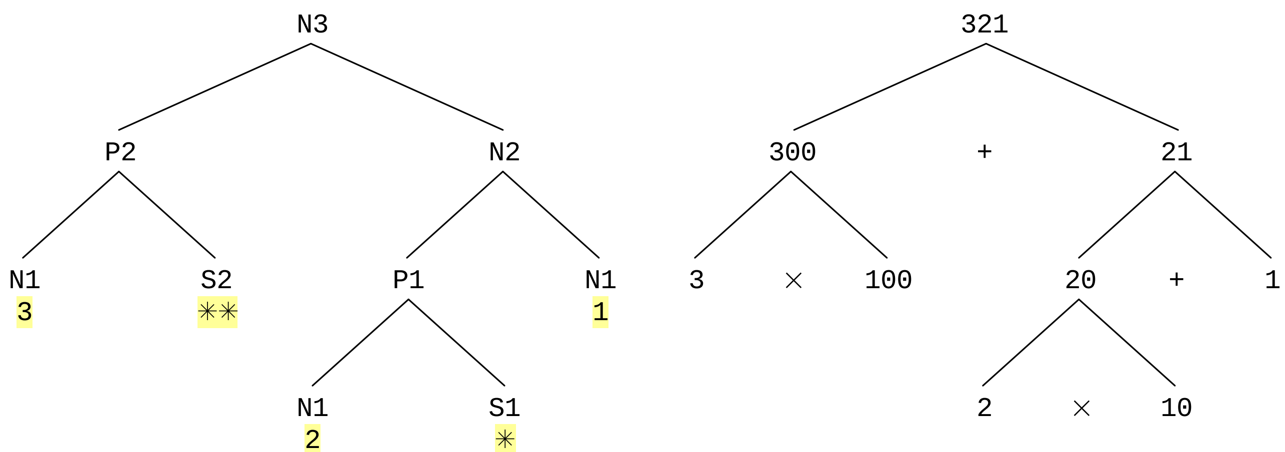
As usual, to provide a semantics, we must provide semantic-values for all the lexical items, and we must provide semantic computation-rules for all branching rules. The proposed semantics for NG3 is given as follows.

(1) $[\![\varnothing]\!] = 0$
$[\![\text{‘0’}]\!] = 0$
$[\![\text{‘1’}]\!] = 1$
etc.

(2) $[\![\text{‘}\ast\text{’}]\!] = 10^1$
$[\![\text{‘}\ast\ast\text{’}]\!] = 10^2$
$[\![\text{‘}\ast\ast\ast\text{’}]\!] = 10^3$
$[\![\text{‘}\ast 6\ast\text{’}]\!] = 10^6$
etc.

(3) $[\![\mathcal{E}_1 + \mathcal{E}_2]\!] = [\![\mathcal{E}_1]\!] + [\![\mathcal{E}_2]\!]$
unless $\mathcal{E}_2$ is an inflectional suffix [S1, S2, etc.], in which case:
$[\![\mathcal{E}_1 + \mathcal{E}_2]\!] = [\![\mathcal{E}_1]\!] \times [\![\mathcal{E}_2]\!]$

By way of illustration, let us consider the syntactic and semantic trees for ‘321’. First, the following depicts the syntactic de-composition of ‘321’.



The following illustrates the semantics in reference to ‘10,101’. In what follows the numerical expressions name numbers, not numerals.

12. A Brief Note On European-Style Numeral Morphology

We have constructed our grammar NG3 using the syntax and semantics of American English, which is based on the following identities.

one thousand	=	10^3
one million	=	10^6
one billion	=	10^9
one trillion	=	10^{12}
etc.		

Unfortunately, the words ‘billion’, ‘trillion’, etc., do not have these meanings in England and the rest of Europe! Rather in Europe, we have the following identities.

one thousand	=	10^3
one million	=	10^6
one thousand million	=	10^9
one billion	=	10^{12}
one thousand billion	=	10^{15}
one trillion	=	10^{18}
etc.		

Accordingly, when we go to construct the grammar for British English, we must go back and redo both the syntactic module and the phonetic module, to take into account this discrepancy. This reconstruction is left as an exercise for the reader.

13. Categorial (Fregean) Reconstruction of NG3 – NG3f

The grammar we have proposed so far – NG3 – provides a compositional semantics for numerals, but it is not categorial/Fregean. In a categorial grammar, all syntactic-composition is functor-application, and all semantic-composition is function-application.

Fortunately, it is fairly simple to reconstruct NG3 in the categorial mold. The central players are the inflectional suffixes (✳, ✳✳, etc.), which we now reconfigure as functors. Specifically, we have the following syntactic categories.

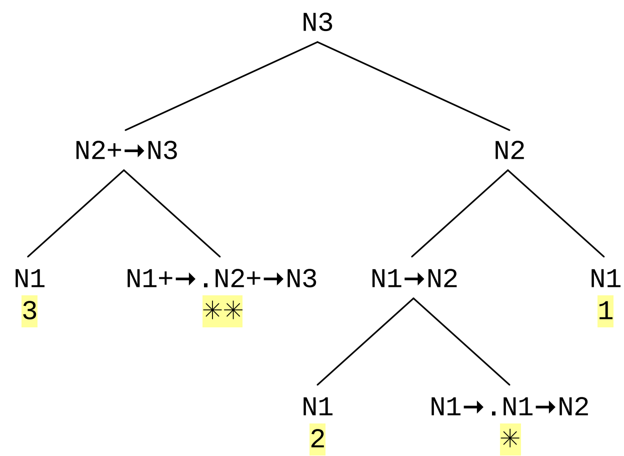
✳1	N1 → (N1 → N2)
✳2	N1 → (N2+ → N3)
✳3	N3+ → (N3+ → N6)
✳6	N3+ → (N6+ → N9)
✳9	N3+ → (N9+ → N12)
etc.	

Here, ‘✳n’ means "n-many stars"; we also have the following definitions.

N2+	$=_{df}$	N2 ∪ N1
N3+	$=_{df}$	N3 ∪ N2+
N6+	$=_{df}$	N6 ∪ N3+
etc.		

This enables us to re-write the syntactic analyses taking into account functional relations. The following is an example.

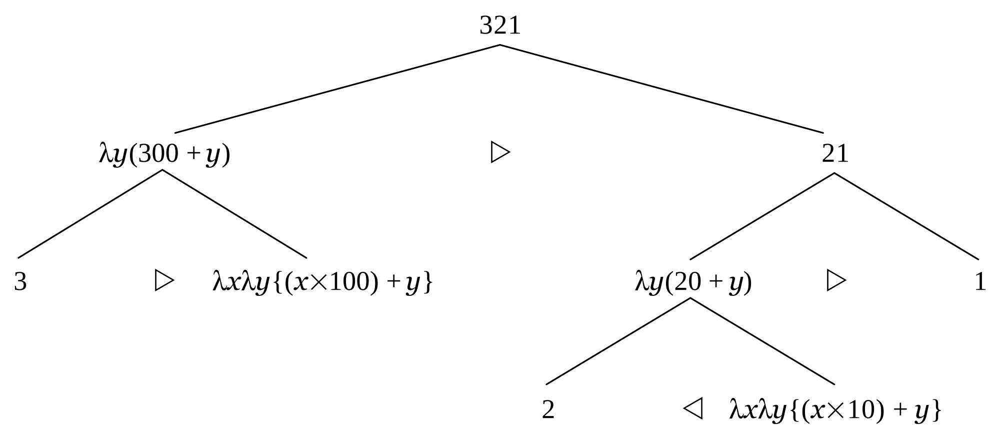
321



On the semantic side, we must provide semantic values for the various functors. This is given by the following clauses.

(1) $[\![\ast 1]\!] = \lambda x \lambda y[(x \times 10) + y]$
(2) $[\![\ast 2]\!] = \lambda x \lambda y[(x \times 100) + y]$
…
(n) $[\![\ast n]\!] = \lambda x \lambda y[(x \times 10^n) + y]$

The following is an example of a semantic-tree.



Note that in this tree the numerical expressions name numbers, not numerals!

14. English Number-Words – ENG1

In the next few sections, we examine the grammar of English number-words (phonograms). As it turns out, we can very easily model the grammar of English number-words on NG3. Indeed, our first proposed grammar – ENG1 – is obtained from NG3 simply by replacing the lexical expressions by their phonetic transcriptions, and making appropriate adjustments in the various modules.

1. Syntax

(1)	N	⇒	N0 | N1 | N2 | N3 | N6 | N9 | N12 | …
(2)	N0	⇒	‘zero’
	N1	⇒	‘one’ | ‘two’ | … | ‘nine’
	N2	⇒	P1 + { ∅ | N1 }
	N3	⇒	P2 + { ∅ | N1 | N2 }
	N6	⇒	P3 + { ∅ | N1 | N2 | N3 }
	N9	⇒	P6 + { ∅ | N1 | N2 | N3 | N6 }
	N12	⇒	P9 + { ∅ | N1 | N2 | N3 | N6 | N9 }
	etc.		
(3)	P1	⇒	N1 + S1
	P2	⇒	N1 + S2
	P3	⇒	{ N1 | N2 | N3 } + S3
	P6	⇒	{ N1 | N2 | N3 } + S6
	P9	⇒	{ N1 | N2 | N3 } + S9
	etc.		
(4)	S1	⇒	‘ty’
	S2	⇒	‘hundred’
	S3	⇒	‘thousand’
	S6	⇒	‘million’
	etc.		

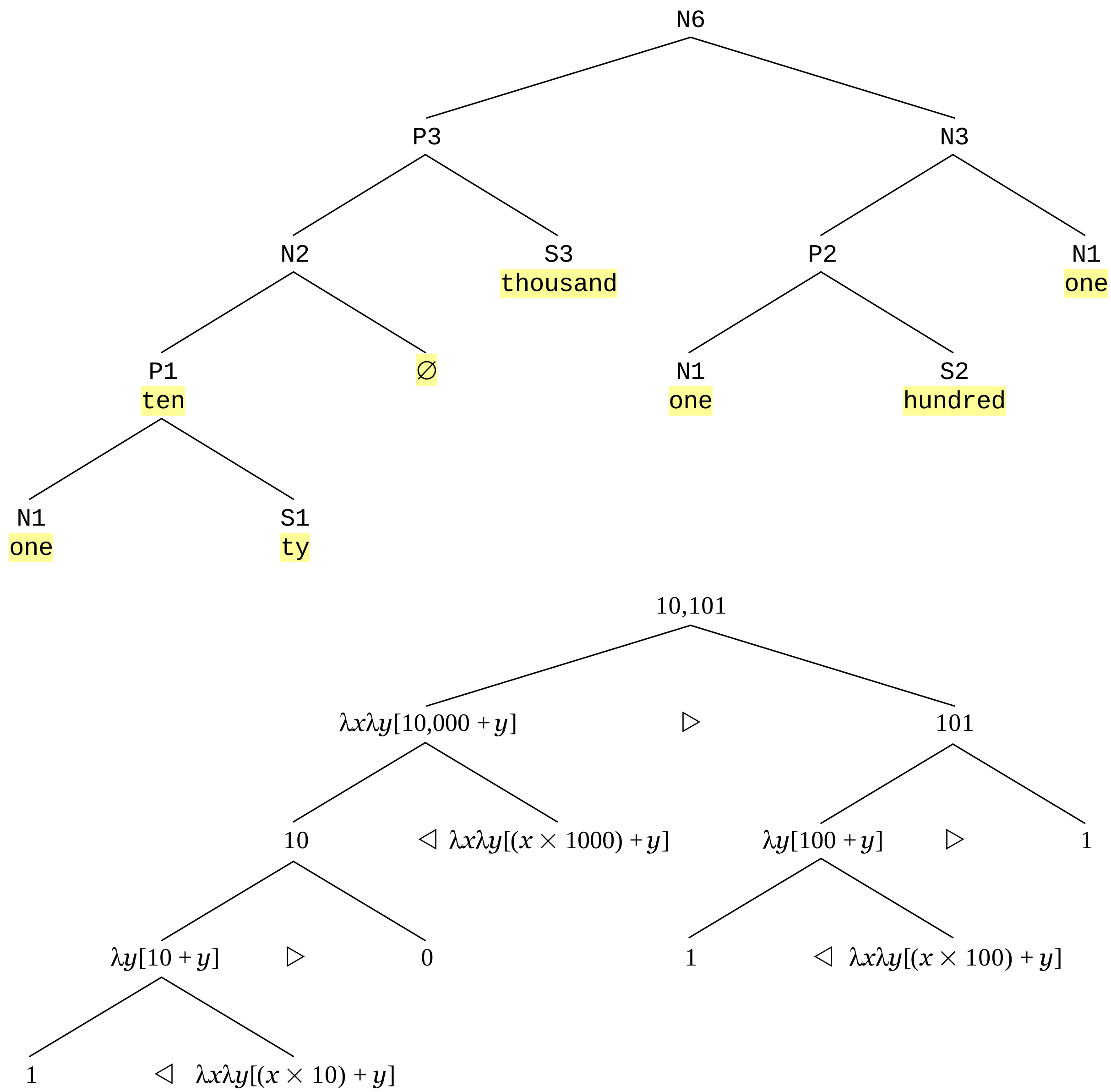
2. Semantics

We can do the semantics of ENG1 pretty much like the semantics of NG3, either in the Fregean or non-Fregean form.

3. Phonology

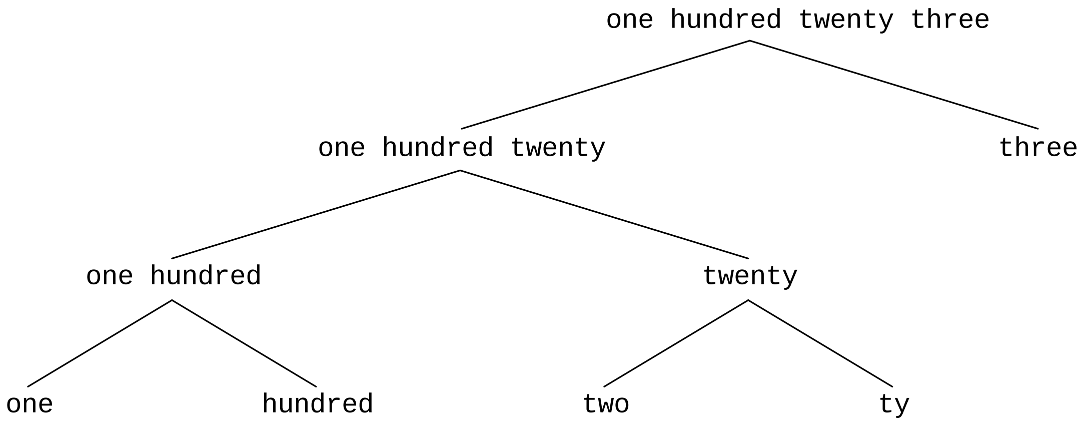
We can likewise do the phonology of ENG1 much like the phonology of NG3. Of course, we are beginning with phonetic items, so the only pronunciation rules will be transformations into colloquial form [e.g., ‘one ty’ ⇒ ‘ten’, ‘two ty’ ⇒ ‘twenty’].

4. Example

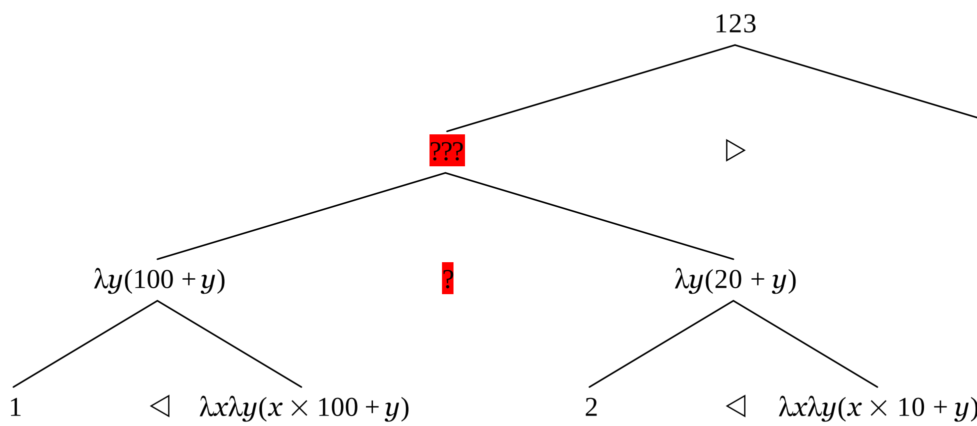

N6
P3
N3
N2
S3
thousand
P2
N1
one
P1
ten
∅
N1
one
S2
hundred
N1
one
S1
ty
10,101
λxλy[10,000 + y]
▷
101
10
◁ λxλy[(x × 1000) + y]
λy[100 + y]
▷
1
λy[10 + y]
▷
0
1
◁ λxλy[(x × 100) + y]
1
◁ λxλy[(x × 10) + y]

15. A Minor Shortcoming of ENG1

So far, we have NG3 and ENG1, which are exactly parallel to each other. They are compositional, and they do not posit invalid grammatical constituents. Nevertheless, they face a minor problem. In particular, it seems that, in their quest to avoid invalid constituents, they go overboard. Consider the following plausible syntactic-tree.



No constituent in this tree appears to be implausible, yet ENG1 and NG3 prohibit this tree, both syntactically and semantically. This is particularly clear in the Fregean variation – NG3F – which *attempts* to build up the associated semantic-tree as follows.



The "fatal error" occurs in the second stage of the construction, where we have two functions neither of which applies to the other.

This problem will be solved in a later chapter, in which we enlarge categorial grammar to countenance a much wider class of semantic compositions.