Chapter 1
Structure and Infinity of Human Language

To begin, we want to understand the capabilities and limitations of some primitive theories of language. Our initial criteria will be that a theory of language needs to have some notion of structure and some way of representing infinity—that is, some way of capturing the fact that there is no longest sentence of English or French or any other human language. Let's start by looking at certain elementary structural properties of human languages.

1.1 STRUCTURE

1.1.1 Structure Dependence

Consider the following pairs of sentences, acceptable to native speakers of English:

(1) a. Susan must leave
    b. Must Susan leave?

(2) a. Bill is sleeping
    b. Is Bill sleeping?

There are many interesting facts about these pairs of examples. One is that for sentences like (a) in (1) and (2) there always exist sentences like (b). Another is that the (a) and (b) sentences in both (1) and (2) "feel related" to all English speakers (importantly, to children as well). Assuming that the acceptability of the (a) sentences may be somehow explained, can we explain why the (b) sentences are possible too?

Sometimes it's claimed that the (a) and (b) sentences in (1) and (2) feel related because they "mean the same thing." I've never understood that claim. Even the most trivial semantic theory tells us that (a) and (b) in each of (1) and (2) do *not* mean the same thing. Further, two sentences that *do* mean the same thing typically do not feel related. Consider the following examples:

(3) a. My father's brother purchased an automobile
    b. My uncle bought a car

These examples certainly do not "feel related," although they presumably mean the same thing. On these grounds, at least, it seems pretty obvious that the phenomenon in (1) and (2) can't be grounded on sameness of meaning.

Let's imagine that there's some process involved in "felt-relatedness" of (a) and (b) in (1) and (2) (such that the language-acquiring child's task is to figure out this process). In particular, suppose (substantially oversimplifying) that this process takes the (a) sentences as basic and does something to them, resulting in the (b) sentences. Notice that there's not yet any empirical or any other motivation for taking the (a) sentences as basic. It might turn out that the (b) sentences are basic, or that something else is basic. For now we simply assume that the (a) sentences are basic to get the investigation off the ground. Given that, what might the process be that we're looking for?

Granting, as standardly assumed, the existence of such things as sentences and words, suppose this process is the following:

(4) Invert the first two words in the (a) sentences.

This works well for (1) and (2), and it refers only to sentences and words (surely irreducible) and, perhaps, "counting to 2." But now consider these examples (the star indicates an example that isn't accepted by native speakers of English):

(5) a. Mary read the book
    b. *Read Mary the book?

(6) a. The man left
    b. *Man the left?

Here we applied (4) and got the wrong results, indicating that (4) isn't the correct hypothesis. In order to pose another hypothesis, we'll need more than just words, sentences, and counting to 2. Suppose that we (and the child) distinguish what kinds of words can be involved in the process in question and what kinds can't be. For the moment let's call *must* in (1) and *is* in (2) *auxiliaries* (without adding any substance to this term yet). Our new hypothesis is this:

(7) Move the auxiliary to the front.

This still works for (1) and (2), and now also for (5) and (6) (i.e., it fails to produce the unacceptable (b) cases, since *read* in (5) and *the* in (6) aren't auxiliaries). As good as it is, unfortunately, (7) still isn't good enough. Consider (8).

(8) Mary has been sleeping

Presumably, (8) has two auxiliaries (if *be*(*en*) is also an auxiliary), but (7) presupposes that the sentence has only one. Changing *the auxiliary* in (7) to *an auxiliary* wouldn't help much, since then (7) would allow an unacceptable counterpart of (8) where the second auxiliary has moved.

(9) *Been Mary has sleeping?

Let's try the following modified hypothesis:

(10) Move the first auxiliary to the front.

This straightforwardly applies to every example that has auxiliaries in it. It seems then to serve as an adequate rule for English "interrogatives." But what about (5a) and (6a), sentences with no auxiliaries at all? One might say that some other process operates there to give a corresponding interrogative sentence. That suggestion isn't ultimately desirable, although for the moment we have to live with it.

Even apart from this limitation, Chomsky has pointed out a type of example that's problematic for the hypothesis in (10).

(11) The man who is here can swim

(11) has two indisputable auxiliaries, *is* and *can*. If, following (10), we move the first auxiliary to the front, our result is (12).

(12) *Is the man who here can swim?

That's complete gibberish.

What's particularly interesting is that we have to look pretty far to find cases where hypothesis (10) fails. For the vast majority of sentences, (10) works fine (modulo the additional process needed to deal with sentences like (5a) and (6a), where the desired results are *Did Mary read the book?* and *Did the man leave?*). And cases lacking the complexity of (11)–(12)

are going to be the entirety of the data that children are likely to be exposed to. So we might expect that children would make mistakes like (12) and then get corrected. But no child has ever been documented making these mistakes. Notably, Crain and Nakayama (1987) did relevant experiments involving elicited production of yes-no questions, which clearly demonstrated that children don't make this kind of mistake (which they, following Chomsky, called a *non-structure-dependent error*). This is a remarkable result. Why don't children make these errors? By far, the most plausible answer is that children are designed so as not to be *able* to make them. When a child is hypothesizing what the question-forming processes are, certain hypotheses are available, others aren't. One that isn't available is the type shown in (10): count the words, label them, and do something with them based on counting. By the same token, no language relates interrogative sentences to declaratives by reading the declarative sentence back to front, by inverting the second and fifth words in the declarative, and so on. A priori, there's no reason why this should be the case (one can easily write a computer language that has these properties). But no human language seems to have these properties.

The right description of the process we're talking about will be stated in terms of *structures*, in terms of hierarchical groupings of words, something like (13),

(13) Move the first auxiliary after the subject to the front.

where the subject can be characterized as a word, two words, three words, and so on, or, more generally, some structure that might be indefinitely long. The right rule is thus *structure dependent*, and, on the face of it, vastly more complicated than the wrong rules rejected above. Chomsky has pretty consistently argued over the years that transformational (displacement) processes of human language never count words or do something to such and such a word. Instead, they are structure dependent: they always look at structures and do something with structured representations.

With the rule in (13), we get the right interrogative form of (11).

(14) Can the man who is here swim?

In order to correctly apply (13), we thus need to identify the subject: *the man who is here*. This then is an argument against viewing sentences as just strings of words, and in favor of viewing them as certain hierarchichal groupings of words. We thus need some procedure that provides a *labeled bracketing*. This procedure will "bracket" some words together and then label the group, so we can then identify the bracketed group of words as a subject.

It will turn out that the label is not really going to be "subject." Chomsky (explicitly in *Aspects of the Theory of Syntax* (1965)) rejects this label for the following reason. In (11) *the man who is here* is certainly a subject. But if we introduce the syntactic category "subject," we will miss an overwhelming generalization: *the man who is here* can also be an object, or an object of a preposition, as in the following examples:

(15) a. I saw [the man who is here]
     b. I talked to [the man who is here]

Similarly for *Susan* in (1) or *Bill* in (2) and so on. It would be a huge coincidence that these same bracketings—these same units of structure—can be found in different positions in a sentence. Thus, we do not want "subject" and "object" to be basic notions in our analysis. Instead, we want those bracketings to have the same label (Chomsky used the term *Noun Phrase* (NP)) wherever they can appear—in the position of subject, object, or object of a preposition—thus capturing the generalization.

Many further phenomena confirm the idea that sentences are structured, that the words in them are grouped into *constituents*.

1.1.2 Constituency Tests

1.1.2.1 Pronominalization

Various tests have proved to be useful in determining what groups of words work together as units of structure, or constituents. One such *constituency test* is whether or not elements can be replaced by pronominal forms.

(16) John left

(17) He left

We might conclude from these sentences that "A pronoun stands for a noun," just as the word *pronoun* suggests. But consider the following cases:

(18) The man left

(19) *The he left

(17) He left

Now, from all of (16)–(19) we might conclude that "A pronoun stands for a noun or a noun preceded by *the*." But look at these examples:

(20) The old man left

(21) The man in the room left

(22) *He in the room left

(17) He left

Adding these cases to the previous ones, we once again need to change our conclusion and say that "A pronoun stands for a noun, or a noun preceded by *the*, or a noun preceded by *the* and followed by..."

Clearly, we're missing a generalization. Somehow, we have to say that *John* has something in common with *the man* and with *the man in the room*, and also with *the old man*, *the old old old man*, and so on. Plausibly, there's some higher-level abstract structural unit, such that the kind of position that *John* fills in (16) can be filled by more complicated elements like *the man* or *the old man* or *the old old old man*.

Following tradition, we will say that *John* is an NP (*noun phrase*, an expression somehow based on a noun), *the man* is an NP, *the man in the room* is an NP, *the old man* is an NP, and *he* is an NP. This is pretty abstract, since *NP* is a symbol that has no phonetic manifestation; we don't pronounce it.

1.1.2.2 Topicalization

A second constituency test involves topicalization. Consider the following pair:

(23) I like John

(24) John, I like

Looking at these sentences, we might conjecture that topicalization is a process whereby we "take a noun and move it to the front of the sentence." Now consider (25)–(27).

(25) I like the man

(26) *Man, I like the

(27) The man, I like

From these sentences, we see that we need to refine our description of topicalization to the following statement: "Take a noun preceded by a definite article, and move both to the front of the sentence.'' But now consider these cases:

(28) I like the old man

(28) *Man, I like the old

(30) The old man, I like

Given these examples, we need to refine our statement even further: ''Take a noun preceded by an adjective (or by adjectives) and a definite article, and move it to the front of the sentence.''

Clearly, we're facing the same type of problem we ran into with pronominalization. How can we characterize the portion of the sentence that undergoes topicalization? It's not just *John*; it's not just *the man*; it's not just *the old man*; it's not just *the old old man*; and so on. It seems, then, that we need a more abstract notion of structure that says that *John*, *the man*, and *the old old man* are the same type of unit.

1.1.2.3 Coordination

A third constituency test involves coordination. Consider the following examples:

(31) I like John and the man

(32) I like John and the old man

(33) *I like John and to go to the movies
(cf. I like John; I like to go to the movies)

Generally speaking (as the ungrammaticality of trying to coordinate *John* and *to go to the movies* shows), only elements of the same type can be coordinated. Therefore, *John*, *the man*, and *the old man* must be of the same type. Why should they be behaving like elements of the same type? How can we characterize this other than in terms of a structural unit?

1.2 INFINITY

Earlier we excluded the possibility that the right answer to the question ''What is 'knowledge of language'?'' could be ''Some kind of a list.'' Why is that? Because we're dealing with individual psychology, and if the answer to the question is ''Some kind of a list,'' then we are saying that that list is in the human brain. If we go on to say that the set of potential behaviors we're interested in characterizing is infinite, then we're saying that there's an infinitely long list in the brain. But how can there be an infinitely long list in the brain? The brain is finite; it's big, but it's finite. So, the simplest formal device that Chomsky was willing to discuss to answer the question "What is 'knowledge of language'?" is the simplest one that can capture infinity. It's called a *finite-state machine/device* (see chapter 3 in *Syntactic Structures*).

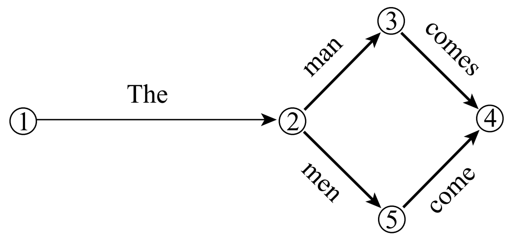
1.2.1 Finite-State Grammars

A finite-state device (or *Markov process*) consists of

- An initial state
- A finite number of states
- A specification of transitions from one state to another (not necessarily distinct)
- A specification of a symbol (or a finite number of symbols) that will be printed when a particular transition obtains
- A final state

Let's look at the simple finite-state machine in (34) (equivalent to (7) on page 19 of *Syntactic Structures*). The circles in (34) represent the five states of the machine. The arrows represent transitions from state to state. ① is designated as the initial state and ④ as the final state. The words above the arrows are the symbols that are printed when a particular transition obtains. So, in (34), upon the transition from the first state to the second state, the symbol *The* is printed.

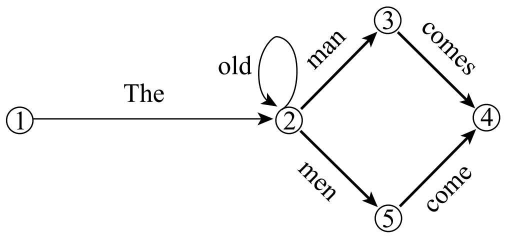
(34)



The finite-state grammar in (34) is the representation of a language consisting of only two sentences: *The man comes* and *The men come*.

Recall that we're interested in infinity. Is there any way that we can characterize infinity with a finite-state device? Let's look at the finite-state grammar in (35) (equivalent to (8) on page 19 of *Syntactic Structures*).

(35)



The grammar in (35) generates the same sentences that the grammar in (34) generated. But it can also generate *The old old man/men* . . . and *The old old old man/men*. . . . We can indeed capture infinity with this machine.

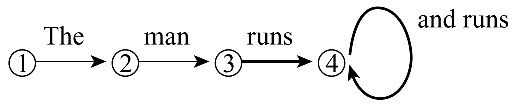
For exposition, and sticking to *Syntactic Structures* terminology, we're temporarily taking a language to be a set of sentences. We're taking a grammar to be a formal machine for generating sentences. A derivation is a sequence of formal computations that results in a sentence (string of symbols).

Let's look at how a finite-state machine will generate a sentence like the following:

(36) The man runs and runs and runs and runs

A machine that generates (36) is shown in (37).

(37)



In addition to (36), the grammar in (37) can generate the following sentences:

(38) The man runs

(39) The man runs and runs

(40) The man runs and runs and runs

The finite-state device in (37) also captures infinity.

1.2.2 Limitations of Finite-State Devices

It's possible to construct some purely formal language and ask whether it's a finite-state language or not. That's what Chomsky does in *Syntactic Structures* (page 21).

Let's look at some formal languages that aren't finite-state languages ((10i) in *Syntactic Structures*). Consider the sentences of the language in (41).

(41) ab
aabb
aaabbb
aaaabbbb
. . .

This language can be characterized as consisting of sentences with any number of *a*'s followed by exactly the same number of *b*'s as long as that number is greater than 0 ($a^n\ b^n$, $n > 0$, using a fairly standard notation).

Can we construct a finite-state grammar for the language in (41)? Let's see if (42) captures this language, whose sentences always contain equal numbers of *a*'s and *b*'s.

(42)

$$\begin{tikzcd}
\textcircled{1} \arrow[r, "a"] & \textcircled{2} \arrow[loop above, "a"] \arrow[r, "b"] & \textcircled{3} \arrow[loop above, "b"]
\end{tikzcd}$$

The language that the grammar in (42) can generate is $a^n\ b^m$, that is, any number of *a*'s followed by any number of *b*'s; but this is different from the language in (41) that we were trying to generate, where a certain number of *a*'s are followed by *exactly the same* number of *b*'s.

The finite-state grammar in (42) *overgenerates*. It can generate all of the sentences of the language in (41), but it also generates many more sentences that aren't part of the language. There's no way to connect up the number of *a*'s with the number of *b*'s. Because (42) has a loop on *a*, it can generate an unbounded number of *a*'s, and because it has a loop on *b*, it can generate an unbounded number of *b*'s; but neither loop can "see" the other loop.

This is exactly what it means for a machine to be a finite-state machine. When it's in a particular state, all it knows is what state it's in, what state it can get to from there, and what symbol it will be printing. It doesn't know what state it used to be in or how many times it's been in that state. The limitation of finite-state machines is that they have no memory—precisely the limitation that prevents them from successfully characterizing the language in (41).

Let's look now at Chomsky's second example, the formal language in (43), called a *mirror-image language* ((10ii) in *Syntactic Structures*).

(43) aa
bb
abba
baab
abbbba
. . .

Let’s see if we can construct a finite-state machine to generate this language. First, let’s try to generate the first part of the sentences in (43). In other words, can we write a finite-state machine to generate any number of *a*’s and *b*’s in any order?

(44)
$$\begin{tikzcd} 1 \arrow[loop above, "a"] \arrow[loop right, "b"] \end{tikzcd}$$

With the finite-state machine in (44) we can indeed generate any number of *a*’s and any number of *b*’s in any order. But now to generate (43) we also have to be able to generate the mirror image of the sentences that we’ve generated with the grammar in (44). We can’t do this with a finite-state grammar, however, because as we just saw, finite-state machines have no memory.

There’s also a human language analogue to the formal languages mentioned above. This example (which I got from Morris Halle, my first linguistics teacher) comes from the realm of word compounding. The military arsenals of many countries have *missiles*. They may also have so-called *anti-missile missiles*, designed to shoot an enemy missile out of the sky. Now, in order to neutralize the anti-missile missiles, the enemy may create something called *anti-anti-missile missile missiles*. There are presumably technological limits on how far one can go in creating this kind of weapon, but there are no linguistic limits as far as the corresponding word formation process is concerned. Let’s notate the situation as follows:

(45) anti^{n} missile^{n+1}
(n *anti*’s followed by $n + 1$ *missile*’s)

As in the case of the formal languages considered above, we can’t characterize this language with a finite-state grammar, and for the same reason: such a grammar can’t keep track of the number of *anti*’s in order to make the number of *missile*’s greater by one.

Finite-state devices have other limitations, as well. Recall that we had proposed the following rule for forming yes-no questions in English:

(13) Move the first auxiliary after the subject to the front.

We've seen that there seems to be no limit on how long the subject of a sentence can be. There's no clear way of characterizing the notion "subject" in terms of number of words. Instead, we have to characterize it with some notion of *structure*. We have to say that *the*, *old*, and *man* in (20), for example, go together and make up some kind of unit. And so far the finite-state device can't capture this. For finite-state machines, sentences are just strings of symbols that get printed on the transitions. This is an obvious limitation of finite-state devices.

Finite-state devices allow us to generate sentences by printing out symbols sequentially but don't determine any abstract structure. We need a more abstract theory that allows us to characterize structure, that allows us to say that words aren't just pearls on a string but entities that can be grouped in natural ways.

It turns out that there's no way to characterize languages like (41) and (43) without introducing abstract structure. Such languages are said to have *unbounded discontinuous dependencies*. The presence of one symbol depends on the presence of another that's an arbitrary distance away in the string, a state of affairs that's beyond the bounds of finite-state description. To overcome this fundamental limitation of finite-state grammars, we will turn to a less elementary, more powerful model that will allow us to describe the languages in (41) and (43) and that will automatically determine abstract structure: phrase structure grammar.

1.2.3 Phrase Structure Grammars

We need a system that allows us to introduce two symbols at the same time, such that these symbols aren't necessarily adjacent to each other. For example, for the "mirror-image language" (43) we need to ensure that when we introduce an *a* at the beginning of a sentence, we're also introducing an *a* at the end of that sentence. We'll see that *context-free* phrase structure (PS) grammars have that power, and with them we can generate languages (41) and (43).

Context-free PS grammars (or *[Σ, F] grammars* in Chomsky's terminology) consist of

- A designated initial symbol (Σ)
- Rewrite rules (F), which consist of one symbol on the left, followed by an arrow, followed by at least one symbol

A PS rule like (46b) will allow us to introduce the symbol *a* at the beginning of a sentence and *at the same time* introduce the symbol *b* at the end of the sentence. The rule allows us to introduce these two symbols simultaneously, without their being adjacent to each other.

(46) a. Σ: S
b. F: S → aSb

Here is what the symbols in (46) stand for:

S	designated initial symbol; also, an abstract, nonterminal symbol that will not be part of any sentence (nonterminal symbols are those that appear on the left side of rewrite rules—hence, are rewritten)
→	a symbol meaning 'rewrite as' or 'consists of'
a, b	terminal symbols (i.e., those that are not "rewritten"—hence, appear in the output sentence)

In contrast, the rewrite rules in a *context-sensitive PS grammar* consist of

- A single symbol on the left
- An arrow
- One or more terminal or nonterminal symbols
- A specification of the circumstances under which the rewriting takes place

For example:

(47) c → d/e ____ f
(or, in other words: ecf → edf)

Many phonological rules in Chomsky and Halle 1968 are context-sensitive rewrite rules. For our purposes right now, however, we will be mainly concerned with rules of the context-free type.

We also need a (recursive) definition of a *derivation*. A derivation consists of a series of lines such that the first line is the designated initial symbol, and the procedure for moving from one line to the next is to replace exactly one symbol by the sequence of symbols it can be rewritten as.

For example, the portion of a PS grammar in (46b) will generate a portion of a derivation that looks like this:

(48) Line 1: S
Line 2: aSb
Line 3: aaSbb
Line 4: aaaSbbb

Notice that no matter how many times we reapply (46b), the resulting string still contains a nonterminal symbol. In order to "turn off the machine" and end the derivation, we must add one more rule:

(49) S → ab

With rule (49) we can rewrite S as *ab* and end the derivation with line (5):

(48) Line 5: aaaabbbb

Thus, the PS grammar incorporating rules (46) and (49) generates the language in (41).

PS grammars allow us to pair up two symbols no matter how much "stuff" is in between. Why? Because the two symbols were introduced by the same application of the same rewrite rule.

The fundamental descriptive advantage of PS grammars compared with finite-state grammars is that PS grammars can pair up elements that are indefinitely far apart. They do this by introducing symbols that are never physically manifested: the nonterminal symbols. We can't turn off the machine until we get rid of all nonterminal symbols. Nonterminal symbols are what we will use to characterize the notion of structure we were concerned about earlier. Why does *John* behave like *the man*, and why does *sleep* behave like *solve the problem*? It's because with respect to the symbols that don't get pronounced (nonterminal symbols), they're the same, even though with respect to the symbols that do get pronounced (terminal symbols), they're wildly different.

We'll look now in detail at how PS grammars represent structure and concentrate on a precise way of doing it.

Let's look at an example of a PS grammar that more closely (though still pretty abstractly) characterizes human language.

(50) a. Designated initial symbol: S
     b. Rewrite rules (F):
        S → NP VP
        NP → N
        VP → V
        N → John
        N → Mary
        V → laughs
        V → sings
        V → thinks

In this grammar the nonterminal symbols (symbols that can be rewritten) are S, NP, VP, V, N;[1] the terminal symbols (symbols that can't be rewritten) are *John*, *Mary*, *laughs*, *sings*.

Here's an example of a derivation using the PS grammar (50):

(51) Line 1: S
Line 2: NP VP
Line 3: N VP[2]
Line 4: N V
Line 5: Mary V
Line 6: Mary laughs
= STOP =

Let's not forget that we need to capture infinity. The PS rules in our little model grammar don't do so. Let's then introduce the following rule (anachronistically, since the *Syntactic Structures* model actually didn't have this type of "recursive" PS rule):

(52) VP $\rightarrow$ V S

Rule (52) is analogous to a loop in the finite-state grammar (recall, e.g., (35) and (37)). It is the "cost" of infinity, which is, in fact, very cheap. Adding rule (52) allows us to generate an infinite number of sentences of the following type:

(53) a. John thinks Mary sings/laughs
b. John thinks Mary thinks John sings/laughs
c. John thinks Mary thinks John thinks Mary sings/laughs
etc.

Thus, by introducing a rule such as (52), which allows us to generate a sentence inside another sentence, we have captured infinity.

Let's now look at one of the many equivalent derivations of one of the sentences in (53), namely, (53a).

(53) a. John thinks Mary sings

(54) Line 1: S
Line 2: NP VP
Line 3: N VP
Line 4: N V S[3]
Line 5: N V NP VP
Line 6: N V N VP
Line 7: N V N V

Line 8: John V N V
Line 9: John thinks N V
Line 10: John thinks Mary V
Line 11: John thinks Mary sings
= STOP =

These rules also generate the following sentences:

(55) Mary thinks John laughs

(56) Mary thinks John sings

However, they generate the following unacceptable ones as well:

(57) *John sings Mary laughs

(58) *John laughs John sings

(59) *John laughs Mary sings

That is, our PS grammar is overgenerating. It's able to generate sentences that aren't part of the English language. The sentences are ‘‘English-like’’ (in word order and structure), but aren't acceptable. Eventually we will address this apparent inadequacy of the model.

PS grammars capture constituent structure by introducing nonterminal symbols. Recall earlier arguments where we concluded that at some level of abstraction *the man*, *the old man*, and *the old old man* all have to be the same type of element. PS grammars allow us to group these phrases under the same nonterminal symbol, NP.

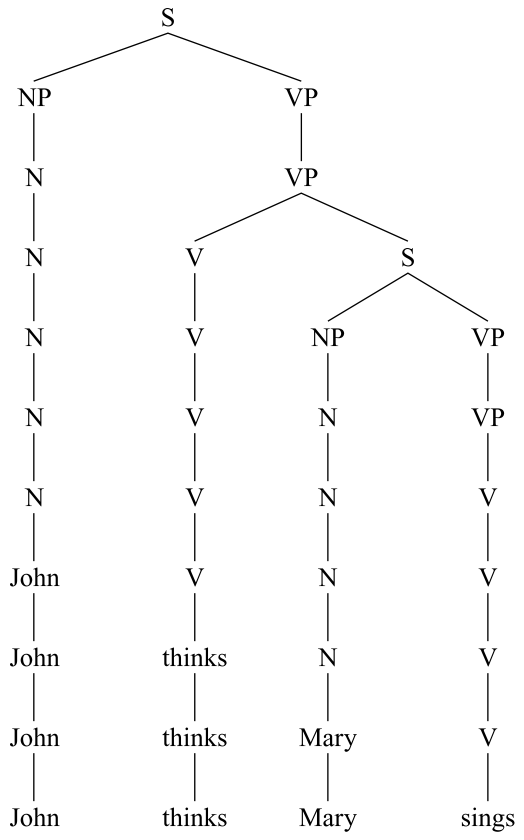
Suppose we take (61), which is one derivation for (60), produced by the grammar introduced in (50) and (52).

(60) John thinks Mary sings

(61) Line 1: S
Line 2: NP VP
Line 3: N VP
Line 4: N V S
Line 5: N V NP VP
Line 6: N V N VP
Line 7: N V N V
Line 8: John V N V
Line 9: John thinks N V
Line 10: John thinks Mary V
Line 11: John thinks Mary sings

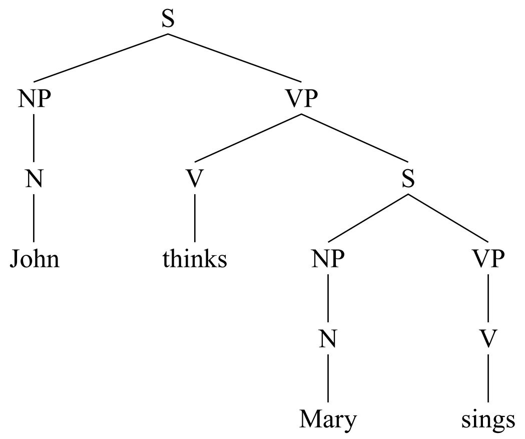
We now connect each symbol with the symbols it was rewritten as. In this way we represent the derivation in the form of a *PS tree*, shown in (62), and we can trace back units of structure.

(62)



Now we can get rid of the symbols that are mere repetitions. In this way, we end up with (63), which is called a *collapsed PS tree*.[4]

(63)



Representing a derivation in this graph-theoretical manner implies a mathematical claim: namely, that no matter which choices we make when we're deriving a terminal string, we will end up with the same collapsed PS tree (abstracting away from structural ambiguity; see below).

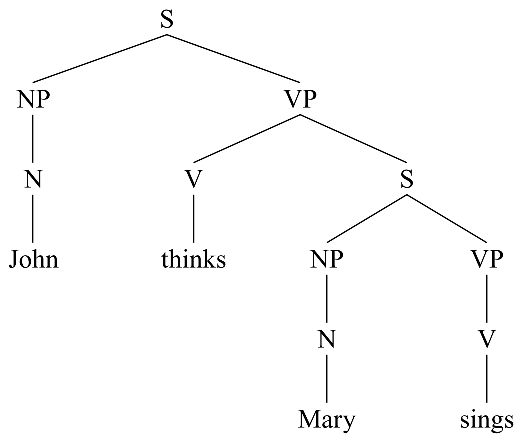
To see this point, let's derive the same sentence in a different way and confirm that we get the same collapsed PS tree.

(64)

Line 1:	S
Line 2:	NP VP
Line 3:	NP V S[5]
Line 4:	N V S
Line 5:	N V NP VP
Line 6:	N V N VP
Line 7:	N V N V
Line 8:	John V N V
Line 9:	John thinks N V
Line 10:	John thinks Mary V
Line 11:	John thinks Mary sings

Indeed, the resulting collapsed PS tree (65) is the same as (63).

(65)



(61) and (64) are two different derivations, but they are in a crucial sense "equivalent." In producing them, we applied the same rules but not in the same *order*. More generally, two PS derivations are equivalent if and only if they involve the same rules, applied the same number of times, but not necessarily in the same order. Recall that we looked at a mechanical algorithm for collapsing a PS derivation down to a PS tree. Indeed, in *The Logical Structure of Linguistic Theory* (1955) Chomsky proved that two PS derivations are equivalent if and only if they collapse down to the same PS tree.

1.2.4 A Note on Recursion

Infinity and structure are arguably the two most fundamental characteristics of human language. We first used finite-state machines to capture infinity; we then changed to PS grammar, a more powerful device that captures both infinity and structure.

At this point I have to admit that (as I hinted earlier) I've been misleading you about one aspect of the theory. Remember Chomsky's formal language arguments for PS grammar. On pretty much that model, I've been analyzing a structure like *John thinks Mary sings* as involving *recursive* PS rules. Rather ironically, however, the theory in both *Syntactic Structures* and *The Logical Structure of Linguistic Theory* (Chomsky 1955; henceforth *LSLT*) didn't have recursion "in the base." Instead, complicated structures were created by special operations, called *generalized transformations*, which put together simpler structures. For example, to derive *John knew that Mary understood the theory*, first the separate structures underlying *John knew it* and *Mary understood the theory* were generated; then a generalized transformation inserted the second of these structures into the first. In other words, in this theory recursion was in the "transformational component." Chomsky listed a few generalized transformations as (22)–(26) on pages 113–114 of *Syntactic Structures*. I will return to this issue and to arguments surrounding recursion in the base versus generalized transformations.

1.2.5 Equivalent versus Nonequivalent Derivations

Consider a vaguely human language–type situation involving equivalent PS derivations and then a vaguely human language–type situation involving nonequivalent PS derivations.

Suppose we have the following grammar, a fragment of one we looked at earlier, where V_N and V_T stand for vocabularies of nonterminal and terminal symbols, respectively:

(66) Σ: S
F: S → NP VP
NP → N
VP → V
N → John
V → left
$V_N = \{S, NP, VP, N, V\}$
$V_T = \{John, left\}$

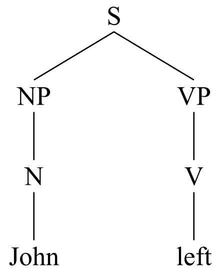
One derivation for *John left* is the following:

(67) Line 1: S
Line 2: NP VP
Line 3: N VP
Line 4: N V
Line 5: John V
Line 6: John left

Another derivation for *John left* is the following:

(68) Line 1: S
Line 2: NP VP
Line 3: NP V
Line 4: N V
Line 5: N left
Line 6: John left

It's easy to verify that the collapsed PS trees corresponding to these two derivations will be the same.

(69)


In addition to what we now know about equivalence of derivations, there's also a clear intuitive sense in which the derivations are equivalent. They're telling us the same information about the constituent structure of the sentence. In particular, (69) contains the following information:

(70) *John left* is an S
John is an N
John is an NP
left is a V
left is a VP

And that's arguably everything there is to know about the structure of this sentence.

Similarly, from the tree in (63)/(65) we can conclude the following information:

(71) *Mary* is an N and an NP
sings is a V and a VP
Mary sings is an S
thinks Mary sings is a VP
John is an N and an NP
John thinks Mary sings is an S

This kind of information is represented as what Chomsky called in *Syntactic Structures "is a" relations*. We needn't worry about the details of this notion at this point. In the next section I'll give an algorithm for determining "is a" relations and discuss its importance for the theory.

Now let's look at a more complicated grammar that allows for *nonequivalent* derivations. Nonequivalent derivations are standardly referred to as exhibiting *structural ambiguity*. Chomsky's classic case of structural ambiguity is (72).

(72) Flying planes can be dangerous

I'll use a slightly simpler example right now.

(73) John hit the man with a stick

Suppose we have the grammar in (74) (as usual, not necessarily an actual fragment of the grammar of English). (In this grammar PP = prepositional phrase, P = preposition, Det = determiner.)

(74) Σ: S
F: S → NP VP
VP → V NP
VP → V NP PP
NP → N
NP → Det N
NP → Det N PP
PP → P NP
Det → a
Det → the
N → John
N → man
N → stick
V → hit
P → with

In *Aspects of the Theory of Syntax* (Chomsky 1965; henceforth *Aspects*), Chomsky called rules like the last seven rules in (74), which rewrite a particular nonterminal symbol as a single terminal symbol, *lexical insertion rules*. In *Syntactic Structures* he made no such distinction among types of PS rules; this lack of distinction severely limited this model's descriptive power, as we'll see later on.

Let's look at two *nonequivalent* derivations of (73). The first is shown in (75).

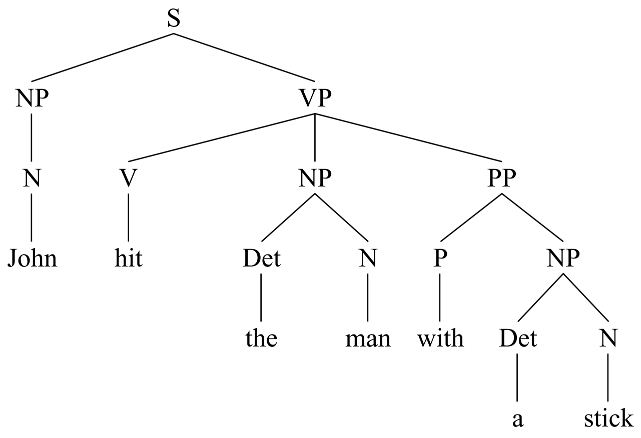
(75)

Line 1:	S
Line 2:	NP VP
Line 3:	N VP
Line 4:	N V NP PP
Line 5:	N V NP P NP
Line 6:	N V Det N P NP
Line 7:	N V Det N P Det N

Line 8:	N V Det N P a N
Line 9:	John V Det N P a N
Line 10:	John hit Det N P a N
Line 11:	John hit the N P a N
Line 12:	John hit the N with a N
Line 13:	John hit the man with a N
Line 14:	John hit the man with a stick

The collapsed PS tree of this derivation is (76).[6]

(76)

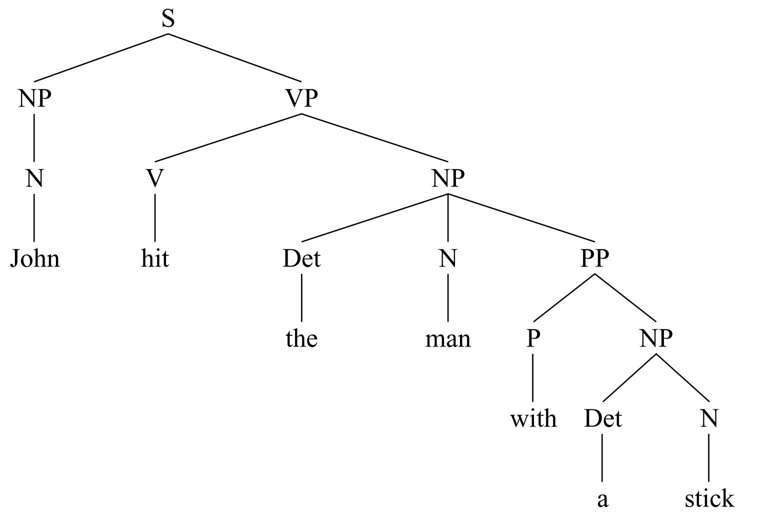


A second derivation and the corresponding PS tree are shown in (77) and (78).

(77)

Line 1:	S
Line 2:	NP VP
Line 3:	NP V NP
Line 4:	NP V Det N PP
Line 5:	N V Det N PP
Line 6:	N V Det N P NP
Line 7:	John V Det N P NP
Line 8:	John hit Det N P NP
Line 9:	John hit Det N P Det N
Line 10:	John hit the N P Det N
Line 11:	John hit the man P Det N
Line 12:	John hit the man with Det N
Line 13:	John hit the man with a N
Line 14:	John hit the man with a stick

(78)



We've generated the sentence we were trying to generate, and we've generated it from the same [Σ, F] grammar we used before. Significantly, though, the derivations (75) and (77) aren't equivalent, because they involve not just a different order of PS rules, but different PS rules. The effect of using different PS rules becomes clear when we collapse the derivations into PS trees ((76) for derivation (75) and (78) for derivation (77)): these two PS trees are different.

Look at what the two different trees (76) and (78) say about the sentence. They both say that *John hit a man with a stick* is an S, that *John* is an N, and that *John* is an NP. They both say that *hit the man with a stick* is a VP, but now they begin to diverge. PS tree (78) says that *the man with a stick* is an NP; however, PS tree (76) says that *the man* is an NP but not that *the man with a stick* is an NP. Let's use two of our constituent structure tests (see section 1.1.2) to see the difference between the interpretations of (76) and (78).[7]

(79) a. John hit the man with a stick (under the interpretation in (78))
     b. John hit him

(80) a. John hit the man with a stick (under the interpretation in (76))
     b. John hit him with a stick

(81) The man with a stick, John hit (has to be the interpretation in (78))

■ *Vukić:* Let's say we have a PS grammar with the following rules:

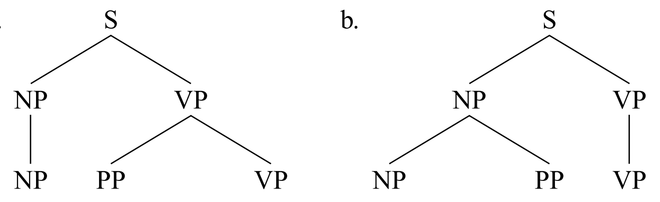
(82) S → NP VP
NP → NP PP
VP → PP VP

Would that introduce an ambiguity? Look at line 3 of (83a) and (83b)—PP VP could be connected to VP, or NP PP to NP (see the corresponding trees in (84)).

(83) a. Line 1: S
Line 2: NP VP
Line 3: NP PP VP

b. Line 1: S
Line 2: NP VP
Line 3: NP PP VP

(84) a. b.



Lasnik: Yes, in principle we can construct such ambiguous cases. This is potentially a problem, since two clearly different derivations can apparently yield the same sequence of lines. One possible answer to this is to construct the tree "on-line." That is, we don't wait until we're done with the whole derivation to construct the tree. Rather, as we proceed from one line to the next, we make the relevant connections. ■

1.2.6 Phrase Markers

1.2.6.1 "Is a" Relations

Recall that Chomsky's theory we are considering is based on

- [Σ, F] grammars (context-free PS grammars)
- Derivations
- Equivalence classes of such derivations
- Set theory

Here is Chomsky's set-theoretic formalization of phrase structure (see *Syntactic Structures*, pp. 27–29, 87):

(85) Given a particular [Σ, F] grammar and a particular terminal string (i.e., string of terminal symbols):

a. Construct all of the equivalent PS derivations of the terminal string.
b. Collect all of the lines occurring in any of those equivalent derivations into a set. This set is the *phrase marker* (PM), a complete representation of the PS of the terminal string.

What do we want a PM to do for us? We want to know for each portion of the terminal string whether that portion comprises a constituent or not, and, when it comprises a constituent, what the "name" of that constituent is.

This theory has an algorithm such that when we apply it, we know exactly the constituent structure of the terminal string and the labeling of the constituents. Specifically, this is an algorithm for determining "is a" relations. (In section 1.2.6.2 we will also discuss a way of simplifying this theory.)

Here is Chomsky's empirical claim: All and only what we want a PM to do is to tell us the "is a" relations between portions of the terminal strings and nonterminal symbols. Anything that tells us those and only those is a perfectly adequate PM; anything that doesn't is inadequate as a PM.

We'll go through an example and see that the resulting PM has some superfluous elements, some things we don't really need. Again, all we need is to determine the "is a" relations. We might build the PM the way we've been doing so far and knock out the excess; the other way is to build it from scratch (Lasnik and Kupin (1977) took the latter course, the simplification I alluded to a moment ago).

Consider the following PS grammar:

(86) Σ: S
F: S → NP VP
NP → he
VP → V
V → left

Now let's construct all the equivalent derivations for the sentence *He left*.

(87) a. *Derivation 1*

Line 1:	S
Line 2:	NP VP
Line 3:	he VP
Line 4:	he V
Line 5:	he left

b. *Derivation 2*
Line 1: S
Line 2: NP VP
Line 3: NP V
Line 4: he V
Line 5: he left

c. *Derivation 3*
Line 1: S
Line 2: NP VP
Line 3: NP V
Line 4: NP left
Line 5: he left

The next step is to collect all the lines occurring in any of these equivalent derivations into a set. That set will be the PM (the PS representation of the terminal string *He left*). Here is our resulting PM:

(88) PM = {$\underline{\text{S}}$, he left, $\underline{\text{he VP}}$, $\underline{\text{he V}}$, $\underline{\text{NP left}}$, NP VP, NP V}

Now the question is, looking at the set above, how can we determine what the "is a" relations are? Some members of the set have exactly one nonterminal symbol and any number of terminal symbols. Let's call them *monostrings* (they're underlined in (88)). By comparing the monostrings with the terminal string one by one, we can compute all the "is a" relations as follows. We start by comparing *he left* with *he VP*.

(89) he left
he VP

From (89), we can deduce that *left* is a VP. Next we compare *he left* with *he V*.

(90) he left
he V

From (90), we determine that *left* is a V. Next we compare *he left* with *NP left*.

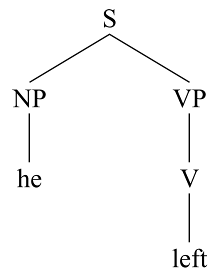
(91) he left
NP left

From (91), we conclude that *he* is an NP. Finally we compare *he left* with *S*.

(92) he left
  S

From (92), we determine that *he left* is an S.

Does that tell us everything we want to know about the "is a" relations in this sentence? Let's look at the tree for *He left*, (93), and see what it tells us about the "is a" relations. We know how to get the tree. We just pick one of the equivalent derivations and collapse it.

(93)


This tree indeed tells us what we've just established.

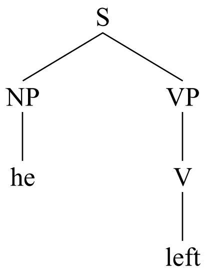
(94) *he left* is an S
left is a VP
left is a V
he is an NP

These are all of the "is a" relations involving the terminal string *he left*. According to Chomsky, an adequate theory of PS is a representation of all and only the "is a" relations that hold in a particular sentence, given a particular [Σ, F] grammar—that is, a representation of what portions of the terminal string bear the "is a" relations to what nonterminal symbols.

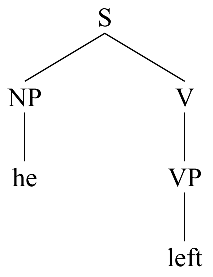
■ *Gutiérrez:* We also have VP left in the tree.
|
V

Lasnik: According to Chomsky, we want the theory of PS to tell us the "is a" relations, where the "is a" relations are defined as relations between portions of the terminal string and nonterminal symbols. The tree we just looked at in (93) actually encodes a bit of information beyond the "is a" relations. The extra information, as you point out, is that this VP was rewritten as V, and the V was rewritten as *left*. Now let's look again at (93), alongside another PS tree, (95).

(93)



(95)



The two graphs (93) and (95) in the graph-theoretic version of PS will collapse into the same set in the set-theoretic representation. Why is that? Because *he V* will be a line in some of the derivations, and *he VP* will be a line in some of the derivations; but the PM is a set, an unordered thing, so there's no way to say that *he VP* comes before *he V*. That's incoherent.

Thus, the graph-theoretic representations encode some information beyond the "is a" relations. The empirical question is, do we need that additional information, say, for phonological, semantic, or further syntactic operations? If we do, then we reject the set-theoretic model; if we don't, then we accept the set-theoretic model, since we would like the minimal theory that does what has to be done. Keep that empirical question in mind as we proceed. ■

We've seen up to now that a PM is a representation of the constituent structure of a sentence. For pedagogical reasons, standard presentations of PMs are never set-theoretic; they're always graph-theoretic. A tree on a blackboard is a much more transparent presentation of structure than a set. But in this theory PMs aren't really graph-theoretic.

1.2.6.2 Reduced Phrase Markers

If all we're trying to do is what Chomsky was trying to do (i.e., determine all and only the "is a" relations), then we have a straightforward algorithm for doing that: namely, look at the terminal string and the monostrings. But in general, a set-theoretic PM will contain much more than the terminal strings and the monostrings. Typically, the monostrings constitute a tiny percentage of all of the strings in the PM.

The question is whether we need all this "extra stuff" or not. Lasnik and Kupin (1977) argued that we don't. Since to determine the "is a" relations we only need the terminal strings and the monostrings, Lasnik and Kupin proposed a construct called a *reduced phrase marker* (RPM), which includes only the terminal strings and the monostrings.

In order to construct an RPM, we could construct a PM in Chomsky's sense and then "knock out" everything except the terminal strings and the monostrings. But Lasnik and Kupin didn't do it this way. Instead, they built RPMs from scratch. They asked, what is a PM in a set-theoretic sense? It's a set of strings. So why don't we start by saying that any set of strings is an RPM, but with some conditions imposed on it. Lasnik and Kupin formalized these conditions: a completeness condition, a consistency condition, and so on. That worked out pretty well and it seemed operationally simpler.

In this model it wasn't necessary to go through all the equivalent derivations to get to the PM. Kupin and I actually worried about that at the time. Here was a theory of PS, but it wasn't based on PS rules at all, totally unlike the classic theory. The job of PS rules is to construct equivalent derivations, but this theory didn't need those equivalent derivations. So the question was, does it make sense to have a theory of phrase structure without PS rules? A few years later a very nice answer emerged, most explicitly in Stowell 1981, where it was argued that PS rules are redundant, duplicating information that must be available in other ways regardless. There's some discussion of this in chapter 1 of Lasnik and Uriagereka 1988.

Exercises

1. Present a finite-state grammar, in graphic form, for each of the following languages:

a. My sister laughed
My sister laughed and laughed
My sister laughed and laughed and laughed
etc.

b. Mary saw three old men
Mary saw three very old men
Mary saw three very very old men
Mary saw three very very very old men
etc.

2. Provide a PS grammar that generates the "mirror-image language" in (43). How many PS rules are necessary and sufficient for this grammar?

3. Given the following context-free PS grammar, give derivations for the sentences *ab*, *aaab*, *aabbbbb*.

$S \rightarrow AB$
$A \rightarrow aA$

A → a
B → bB
B → b

4. Consider the following PS grammar and provide the information requested in A–C.

S → AB
A → Aa
B → b
A → a

A. Provide all of the equivalent derivations for the sentence *aab*.
B. Give the set phrase marker for the sentence *aab*.
C. Give the reduced phrase marker for the sentence *aab* (that is, the phrase marker consisting of the terminal string and the monostrings).

[Remember that in getting from one line of a derivation to the next, exactly one rule applies once. Also, recall that two derivations are equivalent if they differ only in the order of application of rules.]

5. Consider the following languages and provide the information requested in A and B.

a. $a^n bc^n$ (*abc*, *aabcc*, *aaabccc*, etc.)
where n is greater than 0 ($n > 0$)

b. $b^{n+1}a^n$ (*bba*, *bbbaa*, etc.)
$n > 0$

A. Write context-free grammars for each of these languages. (Make the grammars as simple as possible.)
B. Give derivations for two sentences from each of the languages.

1.3 ENGLISH VERBAL MORPHOLOGY

With this much formal background, let's go through some of the English phenomena that Chomsky brilliantly examined in *Syntactic Structures*. We'll begin by looking at a set of complicated facts quite superficially.

1.3.1 English Auxiliaries

In English a sentence can have a main verb and no "auxiliary verb," as in (96).

(96) John sang

A sentence can also have one auxiliary verb, as in (97).

(97) a. John may sing
b. John has sung

c. John will sing
d. John is singing
e. John can sing
f. John could sing

These might be divided into two groups. One group is the following:

(98) a. John may sing
b. John will sing
c. John can sing
d. John could sing

The main verb occurs in its bare form in (98). The second group is the following:

(99) a. John has sung
b. John is singing

The main verb does not occur in its bare form in (99). Other cases in the same group as (99a) and (99b) are these:

(100) a. John had sung
b. John was singing

We'll temporarily ignore the differences in the cases above and analyze all of them as forms of *sing*, which collectively we'll call SING (allomorphs of *sing*).

Now consider the modal auxiliaries. Intuitively, they too are some sort of "helping verb." Syntactically, they all can participate in interrogative inversion: *Must John sing?*, *Could John sing?*, and so on. All of them share the interesting property that when they're followed by the main verb, the main verb is always in its bare form.

Here's a list of modal auxiliaries in English:

(101) may
might
will
would
can
could
must
shall
should

English also has the auxiliary category HAVE, which can take any of the following forms:

(102) has/had/have

And it has the auxiliary category BE, which can take any of the following forms:

(103) am/is/are/was/were

Some sentences have two auxiliary verbs, and in those cases additional forms emerge.

(104) a. John has been singing
b. John could have sung
c. John must be singing
d. John will be singing

A sentence can even have three auxiliary verbs.

(105) John could have been singing

We now have plenty of data. Let's start by formulating some generalizations about auxiliaries; then we'll see how we can formalize those generalizations.

Here's one generalization:

(106) When a sentence contains a modal auxiliary (M), it is always the first thing after the subject.

Here's another:

(107) When HAVE and BE cooccur, BE immediately follows HAVE.

That this is indeed so can be seen from the following examples:

(108) a. John has been singing
b. John had been singing
c. John could have been singing

(We hold constant the fact that there will always be a "main" verb.)

Taking modals and HAVE/BE to constitute a natural class of auxiliaries, we notice the ordering information in (109)–(111).

(109) *Cases with one auxiliary*
M (can occur alone)
HAVE (can occur alone)
BE (can occur alone)

(110) *Cases with two auxiliaries*
a. HAVE BE
   M BE
   M HAVE
b. *BE HAVE
   *BE M
   *HAVE M
c. *HAVE HAVE
   *BE BE
   *M M

(111) *Cases with three auxiliaries*
M HAVE BE

To capture the desired generalization regarding modals and auxiliaries, let's represent a sentence as consisting of the following three parts:

(112) Subject | Aux | Main Verb

That is, we'll say that there's an auxiliary position *Aux*.[8] What can appear in Aux? The following PS rules, in accord with our observations, say that Aux can contain any of the elements shown:

(113) Aux → M
Aux → HAVE
Aux → BE
Aux → M HAVE
Aux → M BE
Aux → HAVE BE
Aux → M HAVE BE

That doesn't look like a generalization at all. Counting the case where there is no apparent auxiliary, there are eight cases and eight rules. What could be worse? How can we possibly make a generalization? Chomsky's proposal in this respect might seem trivial, but it's actually very deep. To fully appreciate it, we need to look at some intellectual background that wasn't made explicit until several years later in *Aspects*.

Throughout his career Chomsky has been concerned with the major questions we're concerned with here: What is our knowledge of language? Where does it come from? How does this knowledge get into our heads? The first question, regarding the nature of linguistic knowledge, Chomsky characterizes in terms of *descriptive adequacy*. A theory of language attains descriptive adequacy if it provides grammars that truly describe the facts of each language. How this linguistic knowledge gets into the head is the question of *explanatory adequacy*. A theory of language attains explanatory adequacy if it provides a procedure by which the learner selects the right grammar on the basis of primary linguistic data (the utterances an infant is exposed to).

In *Aspects* Chomsky proposed a rather precise (if still programmatic) answer to the question of explanatory adequacy. He suggested that certain information is "wired in" the human brain. This information includes the following definitions:

- Definition of a *possible syntactic rule*: This determines what is and what is not a possible syntactic rule.
- Definition of a *possible grammar* (grammar as a set of rules): Not just any set of rules counts as a possible grammar.

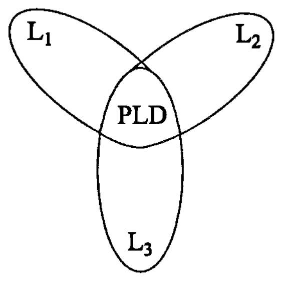
Is this an adequate theory of language acquisition, even if the two definitions are made precise? Well, no. It's the beginning of a theory of language acquisition, but in the *Aspects* model there are just too many possible rules and grammars available to the child. In order to address this problem, Chomsky proposed the following concept:

- An *evaluation metric*

The evaluation metric is a procedure that looks at all the possible grammars compatible with the data the child has been exposed to and ranks them. On the basis of some criterion, it says that G_1 is a more highly valued grammar than G_2, and it picks G_1, even though G_1 and G_2 are both compatible with the data.

For the moment, think of a language as a set of sentences, and think of a grammar as a formal device for generating all and only those sentences. Now, what is the child's task? Born not knowing what the grammar is, and confronted with some primary linguistic data (PLD), the child makes a guess about what the grammar is. And the guess has to be such that it will generate the PLD. It will invariably generate more than the PLD. Why? Because a human grammar has to provide an infinite language. That is the strongest linguistic universal there is. All human languages are infinite. That guarantees that whatever the child's guess is, it will generate vastly more than the (necessarily finite) PLD.

Suppose a child's guess is a grammar G_1 that generates a language L_1 (PLD a subset of L_1). Now, one can imagine a second child being exposed to exactly the same PLD and coming up with a grammar G_2 that is compatible with the PLD and generates $L_2 \neq L_1$. And let's say a third child comes up with grammar G_3 that generates L_3. Suppose that all these languages, an infinite class of them, agree on the PLD and disagree on virtually everything else. This is the most extreme case (to make the point clear), represented in (114).

(114)


So, these grammars are compatible with the PLD (and, by hypothesis, with the formal constraints on possible grammars); hence, all are available to the child as potential grammars. But suppose that G_3 (generating L_3) is the grammar valued most highly by the evaluation metric. Then, when confronted with the PLD, the child is forced to posit G_3, the grammar valued most highly by the evaluation metric that is also compatible with the PLD. This will give the presumably correct result that learners confronted with the same data will acquire the same language; in other words, the situation depicted above, where three children exposed to the same PLD arrive at different grammars, can't arise.

What exactly is the evaluation metric? This is something we don't know a priori (a point that is often seriously misunderstood). It's an *empirical* question. It's the same kind of question as asking what's a possible rule, or what's a possible grammar. Chomsky made proposals about possible rules (he formalized the notion of PS rule, of transformational rule) and a possible grammar (some specific arrangement of these PS rules and transformations). He further proposed that the evaluation metric is this: Given two permissible grammars G_1 and G_2 both compatible with the PLD, G_1 is more highly valued than G_2 if and only if G_1 has fewer symbols than G_2.

With this in mind, let's return to the arrangements of auxiliary verbs. Chomsky proposed in *Syntactic Structures* that there is always at least one item in Aux position, and that that item is there whether or not the structure has any auxiliary verbs. The first item in Aux is always the symbol C (= tense and agreement information; later we'll look at C in much more detail). If so, then instead of the PS rules in (113) we have the ones in (115).

(115) Aux $\rightarrow$ C
Aux $\rightarrow$ C M
Aux $\rightarrow$ C HAVE
Aux $\rightarrow$ C BE
Aux $\rightarrow$ C M HAVE
Aux $\rightarrow$ C M BE
Aux $\rightarrow$ C HAVE BE
Aux $\rightarrow$ C M HAVE BE

Chomsky made the following empirical claim: It's very natural for a language to have the eight rules we've been discussing, rather than some random eight rules for Aux. After all, we could imagine all kinds of different sets of eight rules using these symbols. So, we want to introduce some notation that will allow the evaluation metric to give us the result that the rules in (115) form a natural set. The notation also will be wired in, of course.

Here's the notation that Chomsky proposed:

(116) Aux $\rightarrow$ C (M) (HAVE) (BE)

What does this notation mean? It represents exactly the eight rules we've been discussing, but counted by the evaluation metric as having the cost of only one rule: Aux $\rightarrow$ C M HAVE BE. Informally speaking, M, HAVE, and BE in (116) are each optional.

This move seems seductively natural, but one could imagine that the world could have been otherwise. Imagine a language that has the following rules for auxiliaries:

(117) Aux $\rightarrow$ C M HAVE BE
Aux $\rightarrow$ C HAVE BE M
Aux $\rightarrow$ C BE M HAVE

As far as I know, no language on earth is like that. But maybe Martian is. One may think, then, that Martian is impossible to characterize, because the notation we used for English doesn't work there. But in fact, it could be characterized. We just have to make up a new notation. One natural possibility is this:

(118)
$$\begin{tikzcd}
 & & \text{M} \arrow[dr, bend left] & \\
\text{Aux} \arrow[r] & \text{C} & & \text{HAVE} \arrow[dl, bend left] \\
 & & \text{BE} \arrow[u, bend left] &
\end{tikzcd}$$

By using this notation, we're saying that the three rules in (117) count as no more expensive than any one of them. Why is this a natural notation? Because what comes after C is a set of *cyclic permutations* of M, BE, and HAVE: first pick any one of the elements on the circle, and continue clockwise until all of them are taken.

Mathematically, there's nothing to choose between (116) and (118). A mathematician might want to use the notation in (118); and the study of cyclic permutations is an area of mathematics. It's only *for empirical reasons* that we choose the notation in (116). In other words, the fact that we choose (116) means that Chomsky's notation has nothing to do with mathematical "elegance" or formal possibility; instead, it has to do with empirical facts of human languages.

(116) is the core of the analysis of auxiliaries presented in *Syntactic Structures*. It wasn't until *Aspects* that Chomsky explicitly claimed that the notation is wired in, but the claim was implicit in the earlier work.

The rule in (116) embodies a generalization about the gross order of M, HAVE, and BE. But there are a lot of finer generalizations about how M, HAVE, and BE are actually realized. Recall that M, HAVE, and BE take on a lot of different forms. Even if we know that M, HAVE, and BE occur in the order specified in (116), what tells us which *instances* of M, HAVE, BE—and, finally, V—can actually occur? Here, Chomsky argued that we need something more powerful than just PS rules to account for all of the facts.

1.3.2 Verbal Morphology: More Generalizations

A further look at the examples with which section 1.3.1 began will lead us to the next step. In particular, we will now look more carefully at the finer details of the verbal morphology of English. First let's consider the following alternations:

(119) John sings/sang
John is/was singing
John has/had sung
John can/could sing

Here we seem to have four present/past alternations, holding the subject constant. It's a little tricky to prove that *can/could* is a present/past alternation as opposed to two separate modals. But Chomsky suggested an argument that this is the case, based on a phenomenon often called *sequence of tenses*.

(120) a. John says he owns a house
b. John said he owned a house

(120b) is ambiguous. One reading is that John said, "I own a house"; the other is that John said, "I owned a house." Similarly for the following examples:

(121) a. John said he is tired
b. John said he was tired

(122) a. John says he has solved the problem
b. John said he had solved the problem

When the main verb is past tense, past tense on the embedded verb can be understood as present with respect to the time of the main clause, alongside the expected past tense interpretation. Now note that (123b), with a modal, shows just the same ambiguity.

(123) a. John says he can/will swim
b. John said he could/would swim

So there's some reason for thinking that Chomsky's treatment of modals as showing tense alternations is correct. Thus, we'll assume here that modals do take tense. Ultimately nothing crucial will depend on this treatment, but it's the one that Chomsky gave, so for consistency, and in the absence of a strong counterargument, we'll adopt it too.

Now consider sentences with no auxiliary verbs.

(124) a.	John owns a house	present
b.	John owned a house	past
c.	*John own a house	bare
d.	*John owning a house	progressive
e.	*John owned a house	perfect

Here's the generalization that describes these cases:

(125) *Generalization I*
If the main verb is the first "verblike thing" in the sentence, then it can appear in the "present" or "past" form, but not in the "bare" or "progressive" or "perfect" form.

We have to tease apart "past" and "perfect," which are often phonologically identical. By putting a star on (124e) as opposed to (124b), we're assuming that the "past" and the "perfect" are different. (126) will make this clearer.

(126)	a.	John writes	present
	b.	John wrote	past
	c.	*John write	bare
	d.	*John writing	progressive
	e.	*John written	perfect

The perfect form of *write* in (126e) is different from the past form of *write* in (126b). This is the beginning of an argument to show that the past is generally different from the perfect even though in (124) they have the same phonological form.

■ *Gutiérrez:* Suppose we have a sentence like *I write*. Do we say that this is the bare form? Or do we say that it's the present form, but that the present form and the bare form are phonologically equivalent?

Lasnik: Yes, that's a very important question. Chomsky gave a compelling answer to that question in *Syntactic Structures* showing that there's indeed a "zero morpheme" in a case like that.[9] ■

Can we extend Generalization I to other types of sentences? What's special about a sentence with no auxiliaries is that the main verb is the first "verblike thing." What we want to do is to investigate whether something like Generalization I holds for other things that might happen to be the first "verblike thing" in a sentence. By "verblike thing" we mean, in addition to a verb, a modal or HAVE or BE. For Chomsky, crucially, modals, HAVE, and BE aren't verbs at all—that's why I'm calling them "verblike things."

So let's consider sentences with one auxiliary, starting with HAVE. What forms of HAVE can occur when it's the first "verblike thing" in the sentence?

(127)	a.	John has written	present
	b.	John had written	past
	c.	*John have written	bare
	d.	*John having written	progressive
	e.	*John had written	perfect[10]

The generalization that covers these cases is similar to Generalization I.

(128) *Generalization I′*
If HAVE is the first "verblike thing" in the sentence, then it can appear in the "present" or "past" form, but not in the "bare" or "progressive" or "perfect" form.

What happens when the first "verblike thing" in a sentence is BE?

(129) a.	John is writing	present
b.	John was writing	past
c.	*John be writing	bare
d.	*John being writing	progressive
e.	*John been writing	perfect

The generalization that describes these cases is very similar to Generalizations I and I′.

(130) *Generalization I″*
If BE is the first "verblike thing" in the sentence, then it can appear in the "present" or "past" form, but not in the "bare" or "progressive" or "perfect" form.

Rather trivially, a parallel generalization holds for modals.

(131) a.	John can write	present
b.	John could write	past

(132) *Generalization I‴*
If a modal is the first "verblike thing" in the sentence, then it can appear in the "present" or "past" form and no other form.

Modals seem to trivially fall under Generalization I since they have only present and past forms, and no others, such as a progressive (**canning*), perfect, or bare form. (Bare forms can occur after *to* in the infinitive construction in English, so we can say *to write* or *to be able to*, but we can't say **to can*.)

Putting all these specific generalizations together, we have the following overall generalization:

(133) *Generalization I^+*
Whatever "verblike thing" is the first in a sentence, it will appear in the "present" or "past" form.

Can we find any generalization about the *second* "verblike thing" in a sentence? Consider the following examples:

(134)	a.	He can go	bare
	b.	He is singing	progressive
	c.	He has written	perfect
	d.	*He can goes	present
	e.	*He can went	past

The second "verblike thing" can be progressive, perfect, or bare. But it cannot be past or present. Abstractly, English sentences can have these sequences of two "verblike things":

(135) BE V
M V
HAVE V

The form of the second "verblike thing" depends on what the preceding "verblike thing" is. One combination is "BE + progressive."

(136) a. John is singing
b. John was singing

A second is "modal + bare."

(137) a. John can swim
b. John could swim

A third is "HAVE + perfect."

(138) a. John has written
b. John had written

From these facts, we can deduce Generalization II.

(139) *Generalization II*
a. When a sentence has two "verblike things" and the first is HAVE, then the second appears in the "perfect" form.
b. When a sentence has two "verblike things" and the first is BE, then the second appears in the "progressive" form.
c. When a sentence has two "verblike things" and the first is a modal, then the second appears in the "bare" form.

What happens when a sentence has three "verblike things" and the first is HAVE? As (140) shows, the "verblike thing" that follows HAVE still appears in the perfect form.

(140) a. John has been writing
b. John had been writing

Finally, what if a sentence has more than three "verblike things"?

(141) John could have been writing

The generalization is the same. We can therefore state the following overall generalization:

(142) *Generalization II*$^{+}$
Whenever HAVE occurs, the very next "verblike thing" appears in the "perfect" form.

Now let's look at modals. What kind of thing can come after a modal? (143) gives some examples, and (144) the generalization we can draw from them.

(143) a. John must write
b. John must have left
c. John must be singing

(144) *Generalization III*
Whenever a modal occurs, the very next "verblike thing" appears in the "bare" form.

Finally, let's look at BE. What kind of thing can follow BE? (For the moment I'll abstract away from sentences like *John is tall* and *John was arrested*, which are sentences where BE seems to be the only "verblike thing.")

(145) a. John is sleeping
b. John was sleeping
c. John has been sleeping
d. John may be sleeping
e. John may have been sleeping

The generalization we can draw about BE is the following:

(146) *Generalization IV*
Whenever BE occurs, the very next "verblike thing" appears in the "progressive" form.

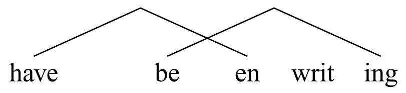
So, when a sentence has BE, it has *ing* (progressive). When a sentence has HAVE, it has *en* (perfect). When a sentence has M, it has the bare form. BE and *ing* do *go* together, but they don't go *together*. There's something in between. HAVE and *en* do *go* together, but they don't go *together*. There's something in between. We have a paradox. The rule that

Chomsky proposed to resolve the first part of the paradox, the dependency, is the following, an extension of (116):

(147) Aux $\rightarrow$ C (M) (have en) (be ing)

As before, this rule is an abbreviation of eight rules. First, as before, C (tense-agreement morphology) precedes the first "verblike thing." Second, this rule correctly guarantees that *have* and *en* are always introduced together, and similarly for *be* and *ing*. But notice that *have* actually introduces the perfect form of the *next* "verblike thing" and *be* introduces the progressive form of the *next* "verblike thing." This situation is shown in (148).

(148)



(148) represents a *cross-serial dependency*. PS rules can't in general deal with these dependencies; they're good at characterizing *nested* dependencies, as we saw, but not cross-serial ones.

To see this point, recall what a derivation is in a PS grammar. It's a sequence of lines, such that the first line is the designated initial symbol, maybe S, we derive the second line from the first by rewriting one occurrence of one symbol, and we keep going until we reach symbols that can't be rewritten. The symbols that can't be rewritten, the terminal symbols, are those that don't occur on the left-hand side of any arrow.

Let's see just how effective a PS grammar is in capturing the generalizations discussed above. Suppose we want to generate the following sentence:

(149) John might have been kissing Mary

We'll be using Chomsky's PS rules (page 111 of *Syntactic Structures*), but we'll add two more rules just to simplify the derivations.

(150) $NP_{sing} \rightarrow$ John
$NP_{sing} \rightarrow$ Mary

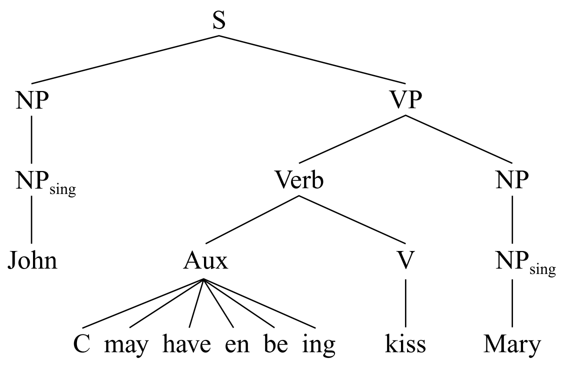
The PS rules give us the following derivation, among numerous possible ones:

(151) S
NP VP
NP Verb NP

NP Aux V NP
NP_{sing} Aux V NP
NP_{sing} C M have en be ing V NP
John C M have en be ing V NP
John C M have en be ing V NP_{sing}
John C M have en be ing V Mary
John C may have en be ing V Mary
John C may have en be ing kiss Mary

We already know that if we collapse all the equivalent derivations, we get the PM for the sentence in (149). Let's look at (152), the collapsed PS tree.

(152)



Apparently, we end up with something that we don't want to generate.

(153) John C may have en be ing kiss Mary

(152) captures the fact that *have* and *en* are introduced together, but not that *en* ends up on *be*. It captures the fact that *be* and *ing* are also introduced together, but not that *ing* ends up on *kiss*. The question is, how can we rectify this? With PS rules we can't generate this sentence in any illuminating way. The best we can do with PS rules is to list all the cases. Chomsky argued on this basis that we need a device more powerful than PS rules, a device that's capable of capturing these paradoxical regularities.