Unit 1

Syntax as a Finite List?

What is the simplest conceivable model of syntax?

Proposal 1.1. Syntax is a finite list of well-formed strings. ⌟

Numerous counterarguments have been proposed, attacking both the finiteness assumption and the feasibility on strings.

Syntax is infinite The usual stuff about unbounded productivity of self-embedding and coordination. This argument is weak because it relies on the competence/performance split. Reject that — and there are no clearcut empirical arguments for this distinction — and the argument falls apart.

Learnability A finite list isn't learnable because learning requires generalization, which in the case of language takes you to an infinite set. Also weak, because: I) every finite language is learnable even under the difficult Gold paradigm, and II) I can reduce the problem of learning finite languages to learning infinite languages. We factorize the finite language F into an infinite language I such that $I \subsetneq F$ and then add a constraint C that limits I to a finite subset.

Succinctness The factorization already hints at the fact that infinite languages usually have more succinct descriptions than finite ones. Here's an analogy for defining sets of natural numbers:

Set	Enumeration	Finite Description
$\mathbb{N}$	$\{0, 1, 2, \ldots\}$	$\{0\}$ closed under successor
$0 \leq i \leq 999$	$\{0, 1, 2, \ldots, 999\}$	$\mathbb{N}$ restricted to first 1000 elements
$0 \leq i \leq 500, i \mod 2 = 0$	$\{0, 2, \ldots, 500\}$	$\mathbb{N}$ restricted to first 250 even elements

But the restriction to a finite subset of an infinite set only adds a constant amount to the description. In computational fields, constant amounts are usually ignored unless they are really high. This is not the case, since the finiteness of natural languages is usually encapsulated by a constraint like "don't have more than 100 levels of self-embedding or coordination".

These arguments only establish that infinity is methodologically useful, but even if we make that assumption it does not commit us to syntax being infinite. The problem

with modeling syntax as a finite list of strings cannot be the finiteness-part. Let's look at the string part then.

Structure It is commonly assumed that sentences contain rich internal structure and that this structure cannot be captured by strings. Technically speaking, this is wrong because a labeled bracketing like $[_{S}\ [_{NP}\ [_{PN}$ John $]]\ [_{VP}\ [_{V}$ slept $]]]$ is obviously a string but contains all the structure we need. But let's interpret the statement in the most gracious manner as targeted against unstructured strings (good luck trying to define that). What is the actual evidence for structure? Taking meaning out of the equation, it is constituent tests. But constituent tests are just generalizations about the set of well-formed expressions: *John and Mary* is assumed to be a constituent because that explains the well-formedness of *John and Mary, Bill likes* in contrast to *John and, Bill likes Mary*. This only shows that constituency is **a** possible way of characterizing the set of well-formed sentences. In order for that to necessarily entail structure, one would have to show that there is no equally succinct alternative. Nobody has ever done that.

Meaning One of the few undeniable facts about language is that sentences have meanings. Parsing is the process of mapping sentences to their meanings, and production is the opposite, mapping an intended meaning to a sentence and uttering that sentence. In contrast to sentences, meaning is obviously structured because of scope and functor-argument relations. We may still encode it as a string, but there is clearly structure there that is much more readily accessible to us than syntactic structure — you do not need constituent tests to see that *John and Bill* form a semantic unit in *Bill likes John and Mary*. As we will see later in the course, meaning is all it takes to derive that syntax is structured. In the words of Steedman (2001):

> "Syntactic structure is no more than the trace of the algorithm which delivers the interpretation."

This computation has a tree-like structure, and that is what syntacticians have equated with the structure of sentences.

Note that the argument for structure is completely independent of whether languages are finite or not. The two are independent claims and should not be lumped together. To further illustrate this, let's look at ways of modelling syntax as infinite string languages and why they ultimately fail without rich structure.