Syntax
Ling 106
February 11, 2002

How do speakers of a language put together a finite number of discrete elements (e.g., words) to generate an infinite number of sentences?

1 Introduction: language, grammar, lexicon

• A language is a set of strings—finite sequences of minimal units—with meaning. The instructions for generating those strings and their corresponding meanings is the grammar of that language. A grammar must specify:

1. a lexicon which contains every minimal unit with meaning (for our purposes, “every minimal unit” means “every word”) and the grammatical category of those units;

2. a syntax, that is, a set of rules for combining minimal units to form longer units, combining these longer units to form yet longer units, and so forth;

3. a semantics, which determines what semantic operation or function corresponds to each syntactic rule.

• Example 1: Propositional Logic

1. Lexicon:
   In this case, the minimal units with meaning are the atomic formulae $p, q, r, s, ...$
   The category specification is simple: Any of these minimal units is a formula of PL.

2. Syntax:
   If $\phi$ is a formula in PL, then $\neg \phi$ is a formula in PL too.
   If $\phi$ and $\psi$ are formulae in PL, then $(\phi \land \psi)$ is a formula in PL too.
   ... etc.
   Nothing else is a formula in PL.

3. Semantics $[\neg \phi] = 1$ iff $[\phi] = 0$.
   $[\phi \land \psi] = 1$ iff $[\phi] = 1$ and $[\psi] = 1$.
   ... etc.
• Example 2: (A fragment of) English

1. Lexicon

<table>
<thead>
<tr>
<th>Grammatical Category</th>
<th>Lexical Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>proper noun</td>
<td>Joan, Bill, Philadelphia</td>
</tr>
<tr>
<td>pronoun</td>
<td>he, she, it, they</td>
</tr>
<tr>
<td>noun</td>
<td>dog, cat, candy</td>
</tr>
<tr>
<td>intransitive verb</td>
<td>walk, sleep, snore</td>
</tr>
<tr>
<td>transitive verb</td>
<td>see, find, hug</td>
</tr>
<tr>
<td>ditransitive verb</td>
<td>give, put, send</td>
</tr>
<tr>
<td>propositional verb</td>
<td>know, claim, believe</td>
</tr>
<tr>
<td>auxiliary verb</td>
<td>will, would, could, must, might</td>
</tr>
<tr>
<td>determiner</td>
<td>the, a, some, every</td>
</tr>
<tr>
<td>preposition</td>
<td>with, in, on, to, before</td>
</tr>
<tr>
<td>adjective</td>
<td>tall, short, green</td>
</tr>
<tr>
<td>adverb</td>
<td>quickly, carefully, very</td>
</tr>
<tr>
<td>complementizer</td>
<td>that, if, whether</td>
</tr>
<tr>
<td>conjunction</td>
<td>and, or, but</td>
</tr>
</tbody>
</table>

2. Syntax:
If $\phi$ is a proper noun and $\psi$ is an intransitive verb, then the sequence $\phi\psi$ (disregarding inflection) is a sentence. Or, more simply: $S \rightarrow N_{pr} V_{intr}$

If $\omega$ is a proper noun, $\phi$ is an auxiliary verb, and $\psi$ is an intransitive verb, then the sequence $\omega\phi\psi$ (disregarding inflection) is a sentence. Or, more simply: $S \rightarrow N_{pr} AUX V_{intr}$

... etc.

Nothing else is a sentence of English.

E.g.
a. Bill snores.
b. Joan might snore.
c. *Joan snore might. (ungrammatical)

3. Semantics
If $\phi$ is a proper noun and $\psi$ is an intransitive verb, then $[[\phi\psi]] = 1$ iff $[[\phi]] \in [\psi]$.

...etc.
2 Memorization

2.1 Lookup table

If knowing the sentences of one’s language were a matter of memorization, each sentence would be associated with a message via a lookup table of some sort. Speakers would simply memorize sentence-meaning pairs.

<table>
<thead>
<tr>
<th>Sentence_1</th>
<th>Message_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence_2</td>
<td>Message_2</td>
</tr>
<tr>
<td>Sentence_3</td>
<td>Message_3</td>
</tr>
<tr>
<td>Sentence_n</td>
<td>Message_n</td>
</tr>
</tbody>
</table>

2.2 Problems

- Novel sentences, infinite number of sentences
- Grammaticality judgments

A genuinely novel sentence, since it is not in the speaker’s lookup table, would have the same status as an ungrammatical string.

(1) a. *Tiger the ate basketball the. (ungrammatical)

b. The tiger ate the basketball. (novel but grammatical)

3 Finite State Automata

3.1 Words and Modes of Combination

- For the sake of the present discussion, the smallest meaningful unit is the word. Speakers learn word meanings plus some method of combining those words to form sentences. Speakers actively compute the meanings of sentences as a function of the meanings of the words in the sentence (i.e., the meaning of the whole is a function of the meaning of its parts).

- What is the simplest device that could compute the method of combining words necessary for natural language?

- We can think of a sentence as a chain of words and a speaker as a device which consists of a finite number of mental predispositions, each associated with a rule that allows the speaker to produce a word and move to a new mental predisposition.

That is, a speaker is equivalent to a finite state automaton (FSA).
This machine generates the following sentences:

(2) a. The boy eats ice cream.
    b. A boy ate hot dogs.
    c. A happy boy ate hot dogs.
    d. One happy girl eats candy.
    e. One happy happy girl eats hot dogs.
    f. A sad dog ate ice cream.

Question: What other sentences does the machine generate?

3.2 Problems

- The FSA approach to computing natural language fails to capture the fact that some words pattern alike.

Words fall into classes defined by intersubstitutivity. Two words belong to the same grammatical category if one can be substituted for the other in a sentence in a way that preserves grammaticality.

(3) a. The happy boy xxxx the girl.
    b. The happy xxxx kisses the girl.
    c. The happy boy kisses xxxx girl.
    d. The xxxx boy kisses the girl.

Recall the chart of grammatical categories from page 2. When people learn language, they do not learn which word follows which other word. Rather, they learn which grammatical category follows which other category.
Finite state automata have a corresponding rule formalism: $A \rightarrow aB$

In the rules below,

1. $A$ is a single symbol (corresponding to a state) called a 'non-terminal symbol'.
2. $a$ corresponds to a lexical item.
3. $B$ is also a single non-terminal symbol.

\[
S \rightarrow \det B \\
S \rightarrow \det C \\
B \rightarrow \adj B \\
B \rightarrow \adj C \\
C \rightarrow \noun D \\
D \rightarrow \text{transitiveVerb} E \\
E \rightarrow \noun
\]

- A finite state automaton has no real memory. So it cannot model sentences that contain items that depend on each other across arbitrarily long strings (called long distance dependencies). And one long distance dependency can be nested in another. To handle all possible long distance dependencies, we would need an infinite number of states, which won’t fit inside a finite brain. See Pinker, chapter 4.

4 Context Free Grammars

- A sentence is not a chain of words, but a tree with hierarchical structure. Words are grouped into phrases which are then grouped into bigger phrases, and so on.

Speakers learn a finite set of phrase structure rules which can generate an infinite set of sentences.

- Context Free Grammar formalism: $A \rightarrow w$

1. $A$ is a single non-terminal symbol.
2. $w$ is a string of terminal symbols (lexical items) and non-terminal symbols. (In the following rules, $\rightarrow$ reads as 'consists of', ( ) indicates optionality, and * means zero or more.)

\[
S \rightarrow \text{NP} \ \text{VP} \\
\text{VP} \rightarrow \text{V} \ \text{NP} \\
\text{NP} \rightarrow (\text{Det}) \ A^* \ N \\
N \rightarrow \text{boy, girl, dog, cat, ice cream, candy, hot dogs} \\
V \rightarrow \text{eats, likes, bites, ate}
\]
A → happy, sad, tall
Det → a, the, one

(4) The happy boy eats ice cream.

Question: What other sentences can the above rules generate?

• Nested long distance dependencies can be handled with recursive rules.
  S → either S or S
  S → if S then S

(5) a.

b.

• Question: Draw some of the tree structures generated by the following rules. How do they illustrate nested dependencies?
  S → aSa
  S → bSb
  S → aa
  S → bb