1 Constituency

1. Constituency tree

```
  John
    \-- thinks
        \   that
           \Suzie
                \-- wants
                    \   to
                       \kill
                           \him
```

2. Labeled constituency tree

```
  S
  \-- John
      \-- VP
          \-- V
              \-- thinks
                  \   C
                      \-- S
                          \-- that
                              \N
                              \-- Suzie
                                  \-- V
                                      \-- wants
                                          \   to
                                             \VP
                                                \-- VP-bar
                                                    \-- V
                                                        \-- kill
                                                            \NP
                                                            \-- N
                                                            \-- him
```
Drawing the tree the way we did first makes a constituency claim, a claim about what the syntactic constituents (= phrases) of the sentence are. In particular, the first tree says *Suzie wants to kill him* is a constituent and the second tree says it is not. The second tree says *that + Suzie* is a constituent, and the first tree says it is not.

Your book discusses various useful tests for constituency. In these lecture notes we talk more some of the underlying motivations for the idea. These include:

(a) Capturing generalizations. This is something all scientific accounts should do. You can think of it as something that guarantees economy of description. But more importantly, perhaps, capturing generalizations mean you can make correct predictions about unseen data.

(b) Capturing distributional patterns. This is really just a special case of capturing generalizations. Distributional patterns are the most basic and in some sense the most important generalizations that syntactic constituency claims capture.
2 Trees

1. A simple tree

```
      S
     /  \
   NP   VP
   /   /  \
 N  V  NP
/   |  |
boys liked N
    |   |
    beans
```

2. Rules that “generate” the simple tree

1. \( S \rightarrow \text{NP VP} \)
2. \( \text{NP} \rightarrow \text{N} \)
3. \( \text{VP} \rightarrow \text{V NP} \)
4. \( \text{V} \rightarrow \text{liked, gave} \)
5. \( \text{N} \rightarrow \text{beans, boys} \)

3. Other sentences “generated” by these rules:

   (a) beans liked boys
   (b) beans liked beans
   (c) boys liked boys

4. These rules were inspired by one sentence, so there are lots of great English sentences not generated by these rules. Here are two sentences NOT generated by these rules:

   (a) boys hate beans
   (b) boys gave boys beans

For each of these sentences, explain why it isn’t generated by these rules, and propose a modification of the rules that would generate it.
5. Adding new rules, modifying old rules

1. NP → Det N New!
2. NP → N
3. Det → the, any New!
4. S → NP VP
5. VP → V NP
6. VP → V NP NP New!
7. V → hate, liked, play, eat Modified!
8. N → beans, boys, harmonicas Modified!

We call a set of rules like this a grammar. This obviously isn’t a grammar in the sense you’re used to using the word. But it is a set of rules that generates a subset of English. We use such partial descriptions to help us study the structures of English. They give us a clear account of how much work our current assumptions about English do.

6. Just a few of the sentences “generated” by the new grammar

(a) boys liked harmonicas
(b) the boys play harmonicas
(c) the boys eat the beans
(d) the boys play the beans
(e) the harmonicas eat the boys
(f) any harmonicas play the beans

Can you think of another? Using a new rule, rule 6? Draw a tree for your new example.

7. Does this grammar generate a finite number of sentences? Why or why not?

3 A grammar

1. Rules: Note: putting X* in a rule means you can have 0 or more X’s in that position. Putting (X) in a rule means you can optionally have an X in that position. So:

   NP → (Det) Adj* N (PP)
means an NP (Noun Phrase) can consist of an optional Determiner followed by any number of Adjectives (including 0) followed by a Noun followed by an optional PP. So the only thing that’s obligatorily present in an NP is a Noun.

1. $S \rightarrow NP \ VP$
2. $NP \rightarrow (Det) \ Adj^* \ N \ (PP)$
3. $VP \rightarrow V \ NP$
4. $VP \rightarrow V \ PP$
5. $PP \rightarrow P \ NP$
6. $V \rightarrow $ liked, hate
7. $N \rightarrow $ fathers, boys, harmonicas, book, beans
8. $Adj \rightarrow $ tall, shiny, new
9. $Det \rightarrow $ the, any
10. $P \rightarrow $ on, in, about, of

2. A tree “generated” by the new grammar:

```
S
  /\    /
 NP  VP
 /\    /
 Det  PP
  \   /
   the fathers
   /\  /\   /
   P  NP  N
    \   /
     of  shiny harmonicas

  \   /
   P  NP
    \   /
     of  N
      \ /
       the boys
```
4. Which of the following sentences are generated by the grammar? For each one that is generated by the grammar, draw the tree that the grammar allows.

(1) a. the tall boys liked the shiny new harmonicas
   b. the shiny boys liked the tall beans
   c. the boys hate any book about beans

5. Fact: This grammar generates an infinite number of sentences. Give some examples of increasingly long sentences. Thinking about this tree may help:

4 Possessives

We are going to revise the grammar to deal with the phenomenon of possessives:

(2) a. John’s father liked beans
b. the king ’s father liked beans

c. the king ’s tall father liked beans

Do as I’ve done here and think of ’s as a separate word. Let’s give it a brand new category Poss. We’ll also think of John as a Noun (N), so our grammar should include the rules

\[
\text{Poss} \rightarrow ’s
\]
\[
\text{N} \rightarrow \text{fathers, boys, harmonicas, book, John}
\]

The first question is, which of the following trees do we want?

(a) \[
\begin{array}{c}
\text{S} \\
\text{NP} \\
\text{Poss} \\
\text{N} \\
\text{V} \\
\text{NP} \\
\text{N} \\
\text{VP} \\
\text{John’s} \\
\text{father} \\
\text{liked} \\
\text{beans}
\end{array}
\]

(b) \[
\begin{array}{c}
\text{S} \\
\text{NP} \\
\text{Poss} \\
\text{N} \\
\text{V} \\
\text{NP} \\
\text{N} \\
\text{VP} \\
\text{John’s} \\
\text{father} \\
\text{liked} \\
\text{beans}
\end{array}
\]

How do we decide? Note: This is a question about constituency claims. One tree claims John ’s father is a constituent. The other does not.

5 Constituency: capturing generalizations

How do we decide between tree (a) and tree (b)?

The question can be rephrased: Is the possessor part of the Subject NP phrase (b) or is it separate (a)? The following fact is relevant:

(3) John ’s father liked John ’s harmonica
In other words, possessors don’t just come before subject nouns; they can also come before object Nouns. Suppose we treated (3) the way we treated possessors in tree (a). We’d have:

(a’)

\[ S \rightarrow (N \text{ Poss}) \text{ NP} \text{ VP} \]

So once again, the possessor (John’s) and the possessed (harmonica) don’t make a constituent. On the other hand, if we treated things as in (b), we’d have:

(b’)

Here are the two grammars we’d need for the two trees, laid side by side, with the rules introducing words left out (because they’re the same)

(a)

1. \( S \rightarrow (N \text{ Poss}) \text{ NP} \text{ VP} \)
2. \( \text{NP} \rightarrow (\text{Det}) \text{ Adj}^* \text{ N} \text{ (PP)} \)
3. \( \text{VP} \rightarrow V \text{ (N Poss)} \text{ NP} \)
4. \( \text{VP} \rightarrow V \text{ PP} \)
5. \( \text{PP} \rightarrow P \text{ NP} \)

(b)

1. \( S \rightarrow \text{ NP} \text{ VP} \)
2. \( \text{NP} \rightarrow (\text{N Poss}) (\text{Det}) \text{ Adj}^* \text{ N} \text{ (PP)} \)
3. \( \text{VP} \rightarrow V \text{ (N Poss)} \text{ NP} \)
4. \( \text{VP} \rightarrow V \text{ PP} \)
5. \( \text{PP} \rightarrow P \text{ NP} \)

Notice the (a) option requires changing the grammar in two places in exactly the same way. That’s not a good sign. It suggests we might be missing a generalization. The (b) option doesn’t have this problem. Now consider a new fact.

(4) John liked the book about John’s father

It looks like we can get possessors inside PPs. How do the (a) and (b) grammars have to be changed to handle these facts? Let’s look at the trees we’d
want to draw.

(a')

On the other hand for (b)

The (a) grammar now changes in 3 places in the same way! The (b) grammar still only changes in 1 place for ALL the facts we’ve looked at.
The (a) grammar is **missing a generalization.** That’s bad! The (a) grammar is missing the following fact:

Wherever you can get NPs you can get possessed NPs.

Here’s the simple intuition. Take any sentence with an NP in it without a possessor. There is another sentence just like it except it has a possessor.

<table>
<thead>
<tr>
<th>John smiled</th>
<th>John’s father smiled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>John liked</td>
<td>John liked Fred’s big black shiny new harmonica</td>
</tr>
<tr>
<td>John read a book about a harmonica</td>
<td>John read a book about Fred’s harmonica</td>
</tr>
<tr>
<td>John gave Fred a harmonica</td>
<td>John gave Fred John’s harmonica.</td>
</tr>
</tbody>
</table>

The (b) grammar captures this simple **distributional** fact: wherever you can get an NP you can get a possessed NP. The (b) grammar does not. (In fact the (a) grammar needs yet another addition to get the last example). You can see it’s missing a generalization because it is stating the same fact multiple times: A possessor is a Noun followed by a Poss; the grammar would be no more complicated if Poss sometimes **preceded** the Noun (say, inside PPs). The (b) grammar says it once and gets all the facts right. Gold star!

## 6 Capturing distribution(al patterns)

The generalization about possessives that the (b) grammar captures is called a **distributional fact** because it is a fact about the distribution of possessives. Distributional facts are the key facts motivating claims about constituency. Some examples of distributional facts and their consequences:

- NP implies possessive: Possessives are parts of NPs
- NP implies Adj: Adjectives are parts of NPs
- NP implies Det: Determiners are parts of NPs

## 7 Recursion: X inside X

So we go with the (a) grammar. But there is still something missing. Consider the following facts:

(5) a. the tall king’s book
b. the tall king’s tall father

c. the king of france’s father

Currently our NP rule reads:

\[ NP \rightarrow (N \text{ Poss}) (\text{Det}) \text{ Adj}^* N (PP) \]

But this data shows we can a determiner to the left of the possessor, and an adjective to the left and to the right and a PP before the Poss. Putting it all together we get:

\[ NP \rightarrow ((\text{Det}) \text{ Adj}^* N (PP) \text{ Poss}) (\text{Det}) \text{ Adj}^* N (PP) \quad (1) \]

Notice any generalization?
The sequence of categories to the left of Poss is exactly the same as the sequence of categories to the right of Poss! In other words, everything that can occur inside an NP (excluding the possessor), can occur before Poss.

In other words, the following rule captures a generalization and keeps us from having to write the same sequence of categories twice:

\[
\text{NP} \rightarrow \text{(NP Poss)} \ (\text{Det} \ \text{Adj}^* \ N \ (PP))
\] (3)

Here are two trees, the first using Rule (2), the second using the improved Rule (3):

**Good**

```
NP
   /\    \
 D - N - PP - Poss - N
   |      |       |       |
 the - king - P - NP - 's - book
   |      |       |
       |       |
       of     N
       |
       France
```

**Better**

```
NP
   /\    \\
 D - N - PP - Poss - N
   |      |       |       |
 NP - Poss - N
   |      |       |
       |       |
       of     N
       |
       France
```

Notice: Once again we’ve made a constituency claim. In *the king of France’s book*, we claim *the king of France* is a constituent, namely an NP.

## 8 Recursion: getting the facts right

The improved rule (3) actually does more than just give us a more economical description; it makes a prediction about unseen data that (2) does not make. This is what capturing generalizations should do.
And it turns out this prediction is right!
Because of course the rule

\[ \text{NP} \rightarrow (\text{NP Poss}) \ (\text{Det}) \ \text{Adj}^* \ N \ (\text{PP}) \]

says anything that can come inside an NP can come inside the Possessor phrase. And that includes possessors. So the improved grammar says you can have possessors INSIDE possessors.

Here is the kind of example that shows that you can:

(6) John liked John’s father’s book

A possessor can itself be possessed!
Notice the bad rule:

\[ \text{NP} \rightarrow (\text{Det} \ \text{Adj}^* \ N \ (\text{PP}) \text{ Poss}) \ (\text{Det} \ \text{Adj}^* \ N \ (\text{PP}) \]

allows at most one Poss per NP, so it can’t possibly handle (6). According to Rule (4), (6) should be ungrammatical. And it isn’t.

9 Properties of recursion

1. **Recursion** is any device in the grammar which allows instances of some category X to occur inside other instances of category X.

2. (5) is called a recursive rule because the category NP occurs both on the left hand side and the right hand side of the rule, so we can get NPs inside NPs.

\[ \text{NP} \rightarrow (\text{NP Poss}) \ (\text{Det} \ \text{Adj}^* \ N \ (\text{PP}) \]

13
3. All on its own, Rule (3) gives us an infinite language. Here’s why:

(7) a. John likes \([\text{NP } [\text{NP John}']\text{’s book}]\).
    b. John likes \([\text{NP } [\text{NP John}']\text{’s father}']\text{’s book}]\).
    c. John likes \([\text{NP } [\text{NP [NP John]’s father]’s father}']\text{’s book}]\).
    d. John likes John’s father’s father’s . . . father’s book.

All of these are grammatical and should be allowed. Recursive rules ALWAYS result in infinite languages.

4. There is another important sense in which our grammar is recursive, because there is another way to get an NP inside an NP:

\[
\begin{array}{c}
    \text{S} \\
    \quad \text{NP} \\
    \quad \quad \text{D} \quad \text{N} \\
    \quad \quad \quad \text{PP} \\
    \quad \quad \quad \quad \text{P} \quad \text{NP} \\
    \quad \quad \quad \quad \quad \quad \quad \text{V} \quad \text{NP} \\
    \quad \quad \quad \quad \quad \quad \quad \quad \text{liked} \quad \text{N} \\
    \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{of} \quad \text{harmonicas} \\
    \quad \text{the fathers} \\
    \text{S} \\
    \quad \text{NP} \\
    \quad \quad \text{D} \quad \text{N} \\
    \quad \quad \quad \text{PP} \\
    \quad \quad \quad \quad \text{P} \quad \text{NP} \\
    \quad \quad \quad \quad \quad \quad \quad \text{V} \quad \text{NP} \\
    \quad \quad \quad \quad \quad \quad \quad \quad \text{liked} \quad \text{N} \\
    \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{of} \quad \text{harmonicas} \\
    \quad \text{the fathers} \\
\end{array}
\]
Moral: Recursion in the grammar ALWAYS guarantees the grammar has an infinite number of sentences, even this kind of recursion, which is called **indirect recursion**

5. There are other ways to get an infinite number of sentences that aren’t recursive. For example, even before we introduced recursion, our grammar already used another device that guaranteed it generates an infinite number of sentences. Consider this simplified version of the rule we had before we introduced possessives:

\[ NP \rightarrow (\text{Det}) \text{ Adj}^* \text{ N} \]

Look closely at this simplified rule. We took out the recursion for possessives, and we took out the PPs, which introduced indirect recursion. But this rule still “generates” an infinite number of NPs. *How does this rule generate an infinite number of NPs?* Give examples of longer and longer NPs generated by this rule.

Notice that even though we get an infinite number of NPs, we still don’t get NPs inside NPs this way. So this isn’t recursion. This device, using the \(^*\), is called, **iteration**.

6. There are important computational differences between recursion and iteration, and it isn’t the business of this course to go into them. But the key idea is this: **Recursion requires an arbitrary amount of memory to process**: The deeper the recursion in the example you are processing (the more NPs nested inside other NPs), the more memory it takes to process the example. Iteration, on the other hand, is inherently simpler. You have to allow an infinite number of \(^*\)ed items, but you don’t have to remember how many you’ve seen to process what’s coming next correctly, so as you process each one, you can forget about the last one; therefore, a processor with a finite fixed amount of memory can do the job.

## 10 Center-embedding

We know we need recursion to describe human languages (proving this rigorously is actually more complicated than we’ve let on here). But the interesting paradox is this: it appears that **in certain circumstances**, humans have
a lot of trouble maintaining the arbitrary amount of working memory that recursion requires. The sentences that demonstrate this exhibit a property called center embedding. Here are some examples. To begin with, here is a classic nursery rhyme, as recalled by Yngve (1960):\(^1\)

```
This is the dog,
that worried the cat,
that killed the rat,
that ate the malt,
that lay in the house that Jack built.
```

Now contrast this with a slightly rearranged version, where we start with the malt, and get ready to run out of working memory:

(8) a. This is the malt that the rat ate.

b. This is the malt that the rat that the cat killed ate.

c. This is the malt that the rat that the cat that the dog worried killed ate.

Did you run out of memory? Notice the hard-understand parts interrupt a sentence you are in the middle of processing:

```
This is the malt
[ s that the rat [ s that the cat [ s that the dog worried ] killed ] ate ].
```

They are embedded in the center. Hence, center-embedding. When we get

\(^1\)In another version, the nursery rhyme is longer and a little more fun:

```
This is the farmer sowing the corn,
that kept the cock that crowed in the morn,
that waked the priest all shaven and shorn,
that married the man all tattered and torn,
that kissed the maiden all forlorn,
that milked the cow with the crumpled horn,
that tossed the dog,
that worried the cat,
that ate the malt,
that lay in the house that Jack built.
```

Note: if you have trouble with *lie* and *lay* you may have trouble with this rhyme, in which intransitive *lay* occurs in the past tense (http://grammar.quickanddirtytips.com/lay-versus-lie.aspx).
to *ate* it’s a real effort to remember who’s doing the eating (the rat!) and what’s getting eat (the malt!). Notice the difference when we start with the dog

This is

the dog [s that worried the cat [s that killed the rat [s that ate the malt]]].

We still go 3 S’s deep, and it’s still recursion (S inside S); but the S’s aren’t stacking up in the middle of an S we are processing. They’re at the edge. And that clearly makes a difference. When we get to *eat*, the malt is right there, and in each case, it’s the most recently processed Noun Phrase (for example, *the rat*) that does the eating.

Yngve also cites other examples of center-embedding:

(9) a. That it is obvious isn’t clear.
   b. That that it is true is obvious isn’t clear.
   c. That that [that they are both isosceles is true] is obvious isn’t clear.

Compare the last sentence to the following paraphrase, which doesn’t scramble your brain:

(10) It isn’t clear that it is obvious that it is true that they are both isosceles.

Again, in the easier to understand version, the S’s stack up at the edge. And:

(11) a. What it would buy in Germany was amazing.
   b. What what it would cost in New York would buy in Germany was amazing.
   c. What what what he wanted would cost in New York would buy in Germany was amazing.

Compare the last sentence to:

(12) It was amazing what could be bought in Germany for what what he wanted cost in New York.
References