Overgeneration and the Lexicon

1 Complementizers and null categories

We began by introducing some changes to our phrase structure rules to allow for complementizers such as *that, if, whether* and the null complementizer (which will be referred to as “0” here). We also introduced null determiners and null auxiliaries (also written as “0”). The new PS rules are:

\[
\begin{align*}
S & \rightarrow NP \text{ Aux } VP \\
CP & \rightarrow C \ S \\
VP & \rightarrow V \ (NP) \ (CP) \\
VP & \rightarrow VP \ PP \\
NP & \rightarrow Det \ Adj^* \ N \\
NP & \rightarrow NP \ PP \\
PP & \rightarrow P \ NP
\end{align*}
\]

At the moment, the lexicon looks like this:

\[
\begin{align*}
N & \text{ John} \\
N & \text{ man} \\
\ldots
\end{align*}
\]

\[
\begin{align*}
V & \text{ like} \\
V & \text{ think} \\
\ldots
\end{align*}
\]

\[
\begin{align*}
C & \text{ that} \\
C & \text{ if} \\
C & \text{ whether} \\
C & \text{ 0} \\
\ldots
\end{align*}
\]

\[
\begin{align*}
\text{Det} & \text{ the} \\
\text{Det} & \text{ every} \\
\text{Det} & \text{ a} \\
\text{Det} & \text{ 0} \\
\ldots
\end{align*}
\]

\[
\begin{align*}
\text{Aux} & \text{ should} \\
\text{Aux} & \text{ has} \\
\text{Aux} & \text{ might} \\
\text{Aux} & \text{ 0} \\
\ldots
\end{align*}
\]

\[
\begin{align*}
P & \text{ with} \\
P & \text{ on} \\
\ldots
\end{align*}
\]
Some example trees using these new rules:

```
S
  NP  Aux  VP
  |    |    |
  Det N  V  NP
0 Bill  read the book

S
  NP  Aux  VP
  |    |    |
  Det N  has V  NP
the man seen 0 Mary
```

“Bill read the book.”

“The man has seen Mary.”

```
S
  NP  Aux  VP
  |    |    |
  Det N  V  CP
0 John  thinks that

S
  NP  Aux  VP
  |    |    |
  Det N  has  NP
that has seen Det N
0 Mary seen some guy
```

“John thinks that Mary has seen some guy.”
“John thinks that Mary thinks the man likes the woman.”

2 Lexical items as bundles of features
After introducing these changes to the PS rules, we considered the problem of overgeneration: our rules generate far too many bad sentences:

* John kissed that Mary likes Bill.
* The boys kisses Mary.
* John persuaded that Mary likes Bill.
Etc. etc.

Before facing up to this problem, we investigated the nature of the lexicon. Up to this point, we’ve assumed that the lexicon is just a list of words and categories (“dog is an N”, “think is a verb”, etc. etc.) Now we’re going to get more sophisticated and start thinking about lexical items (words) as bundles of features.

In fact, we’re already done the same things for sounds. For example, the sound /d/ can be understood as the following bundle of features:

\{ place(alveolar), manner(stop), voicing(voiced) \}
For words, we need to figure out what kinds of features they can have and what values those features can have. This will be different for words of different categories. Starting with nouns, we have something like the following:

### Nouns:

<table>
<thead>
<tr>
<th>Name</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>N</td>
</tr>
<tr>
<td>Number</td>
<td>singular/plural</td>
</tr>
<tr>
<td>Class</td>
<td>count/mass/proper-name</td>
</tr>
</tbody>
</table>

This is oversimplified and leaves many features out, but it’s good enough for now.

The table above gives us a *feature template* for nouns – all nouns must have these features. The template for verbs is as follows:

### Verbs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>V</td>
</tr>
<tr>
<td>Number</td>
<td>singular/plural</td>
</tr>
<tr>
<td>Tense</td>
<td>past/present</td>
</tr>
<tr>
<td>Subcategorization</td>
<td>___CP/___NP/___NP CP/etc.</td>
</tr>
</tbody>
</table>

The only surprise here should be the “subcategorization” feature. This feature tells us what kind of object the verb wants. Verbs like *eat*, for example, must have NP objects (*“John ate the apple”* is OK but *“John ate that Mary likes Bill”* is not). Other verbs, such as *think*, require CPs (*“John thinks that Mary likes Bill.”*) There are even some verbs such as *tell* which require both (*“John told Bill that Mary likes Jane.”*) The subcategorization feature of a verb gives us this information. As an example, here are the subcategorization features for some of the verbs we’ve looked at:

- think   ___CP
- eat     ___NP
- tell    ___NP CP

Note that some verbs can take more than one kind of object. For example, the verb *think* can have an NP object as well as a CP object (*“John thinks the same thing.”*) However, to keep things simple, we are just going to ignore this fact and assume that every verb has exactly one subcategorization feature.

### 2.1 Features of N and features of NP

NPs have the same features as the nouns inside them. For example, the NP “the boys” is plural because the noun inside it (“boys”) is plural. Similarly, VPs have the same features as the verbs inside them. For example, the VP “hits Bill” is in the present tense because the verb “hits” is in the present tense. In general, we can say that:
If X has the feature F, then XP also has the feature F.

2.2 **Context sensitive and context insensitive features**

Some features of a lexical item are independent of the context in which the lexical item appears. For example, a noun can be singular or plural in any context. Other features are context sensitive. For example, a verb can only have a “-s” ending if it is preceded by a singular subject NP:¹

The boy likes Mary
* The boys likes Mary

To put this in terms of features, whereas the subject NP can have whatever number feature it likes, the verb can only have the feature \{number(singular)\} if the subject NP also has this feature. This distinction between context-sensitive and context-insensitive features will become important shortly.

3 **Lexical insertion**

*Important: you will not be expected to remember the following material in detail. This is just for people who are curious to see exactly how the system works. The general point to bear in mind is that features on one node in a tree can depend on features in another node in the tree. There are rules like “If the subject NP has the feature \{number(singular)\}, then the verb must have this feature too.” You should be able to understand rules like this when they are written in ordinary English.*

¹ This is an oversimplification: more precisely, it can have the “-s” ending if it is in the present tense and has a 3rd person singular subject, but we will ignore these complications.
We need to figure out how to get from a tree like (A) to a tree like (B):

(A)

(B)

The only difference between the two trees is that words have been inserted under the categories Det, N Aux and V in (B). This process of inserting words is called *lexical insertion*. Clearly, we have to make sure that the right kind of word gets inserted in the right place. For example, we don’t want to insert the noun “food” under V.

The first step in lexical insertion is to replace the symbols Det, N, Aux and V in the tree in (A) with bundles of features which match the feature templates for these categories. So for example, we could choose to replace the first N with the feature bundle \{category(N), number(singular)}\}. Of course, we could just as well choose the bundle \{category(N), number(plural)}\} instead. It’s important that we only include *context-insensitive features at this point*. Since we haven’t figured out the feature templates for Det and Aux yet, we will just write \{category(Aux),…\} and \{category(Det),…\}. In other words, we know that they have the features \{category(Aux)}\} and \{category(Det)}, and the “…” indicates whatever other features we’ve missed out.

A final thing to remember is that NP will be replaced with the same feature bundle as N, and that VP will be replaced with the same feature bundle as V. (See section 2.1)

Once all this is done, we get the following tree:
I've used “cat” as a shorthand for “category” and “num” as a shorthand for “number.”

Now it’s time to deal with the context-sensitive features of V. What we basically want to say is that V gets assigned a subcat feature based on the category of what’s next to it within the VP. So for example, if there is something with a {category(N)} feature to the right of the V, the V gets assigned the subcat feature {subcat(__N)}. We can state this more formally using the following rule (don’t worry if you don’t understand this bit):

\[
\{\text{category(V)}\} \Rightarrow \{\text{subcat(X\_Y)}\} / \{\text{category(X)}\} \_\_ \{\text{category(Y)}\}
\]

With this rule in place, we can finally add a subcat feature to the verb:

As shown in the tree, the verb has received the feature {subcat(__N)} because the node to the right of it in the tree has the category N. Now we can consult our lexicon and see which words have the right features to be inserted in each node. The most interesting case is that of the verb.

**Lexicon:**

...  
**kisses**  \{category(V), tense(present), number(singular), subcat(__NP)}  
**thinks**  \{category(V), tense(present), number(singular), subcat(__CP)}  
...

As can be seen, **kisses** has a subcat feature compatible with the one in the tree, but **thinks** does not. Therefore we prevent our grammar from allowing sentences like “John thinks an apple.”

We will extend this system to cover some other cases, like subject/verb agreement. To give a preview, the rule for establishing subject/verb agreement is as follows:
\{\text{category}(V)\} \Rightarrow \{\text{number}(X)\}
/ \{\text{category}(N),\text{number}(X)\} \{\text{category}(\text{Aux})\} ___

In other words, a node of category V which is preceded by an NP and an Auxiliary must have the same number feature as the N. This rule will assign the number feature to the VP (since it’s the VP that’s preceded by an NP and an Aux), and because V and VP always have the same features, the verb will also receive this number feature.