X-bar theory continued…

(1)

```
XP
   /
  /  
Spec  X'
      /
specifier

YP  X'
   /  
  adjunct
X  ZP
     
head  complement
```

Substitution and adjunction

Two kinds of movement: **Substitution** and **Adjunction**.

**Substitution**: Replaces an empty (but existing) position with a moved element.

Passives:
(2) a. \([\text{IP} \ e \ [I \ [\text{VP} \ was \ [\text{VP} \ solved \ the \ problem \ ]]]]\)
    b. \([\text{IP} \ the \ problem, \ [I \ [\text{VP} \ was \ [\text{VP} \ solved \ t_i \ ]]]]\)

**Wh-questions**:
(3) a. I wonder… \([\text{CP} \ e \ [C \ C_{[+Q]} \ [\text{IP} \ John \ bought \ what \ ]]]]\)
    b. I wonder… \([\text{CP} \ what_i \ [C \ C_{[+Q]} \ [\text{IP} \ John \ bought \ t_i \ ]]]]\)

**Adjunction**: Creates a new position by attaching to an existing node.

(4)

```
XP
   /
  /  
ZP_i  X'
      /
X  YP
     
...t_i...
```

(5) a. \(\text{XP} \)
    b. \(\text{XP} \)

```
XP
   /
  /  
Spec  X'
      /
X
     
...  
```

```
Spec  X'
      /
ZP_i  X'
     
...  
```
In (5a) there is only one real XP node; in (5b) there is only one real X′ node. The fact that two are written is a convention. Maybe closer to this:

(6) \[ \text{XP} \]

(although: we will later need to make use of the notation convention used in (5))

Recall the tree concepts of dominance and sisterhood.
In (6) and (5a): Spec and X′ are sisters. ZPi has no sister.
XP dominates ZPi, Spec, X′ (and X and …)

In (4), ZPi and X′ are sisters.
XP dominates ZPi, X′ (and X and YP)

Topicalization:

(7) I know…

Extraposition and Heavy NP shift are also adjunction movements, right-adjoining to VP.

Head movement (movement of terminal categories) is also adjunction, where the moving head adjoins to the target head.
Speaking of adjunction… We can also think of attachment to $X'$ this way:

(9)  
```
      VP
     /   \
    V'    PP
   /    \   into the net
V'   Adv
    /    \
 V  V'
```

Where we are

X-bar theory says that trees are all built on this template (for any category $X$):

(10)  
```
      XP  \-
     /   \  maximal projection
specifier \-   XP  X'  \-
    head  X  ZP  complement
```

We also have the ability to *adjjoin* phrases to each level of the representation.

(11)  
```
      XP  \-
     /   \  adjunction to XP
adjunct  XP  \-
Spec X'  adjunction to X'
adjunct  X'  adjunction to $X^\circ$
X  complement
Y  X  adjunction to $X^\circ$
```

Adjunction allows for iteration (whereas there is only one specifier and one complement).

As a result of the changes we’ve made, most of the trees have only binary branches. We can take this a step further and require that our trees have no more than binary branches:

**Binary Branching**

A node can dominate at most two branches.
Note: This causes a problem with ditransitive verbs like put, which requires that we re-think how they work, but we’ll hold off.

We are looking for a maximally simple, restrictive framework that can describe all and only the structures of sentences of natural language. X-bar theory is much better than PS rules—primarily, it is much more restrictive. There was very little constraint on the kind of PS rules we could write—we could write PS rules describing any structure we see in natural language, but also for lots of structures we would never see.

---

**Small clauses**

(12) a. John considers [IP [NP Bill ] [I to [VP be [AP incompetent ]]]].  
    b. John considers Bill incompetent.

What is the structure of the second one?

There’s no to, and there can be no be, suggesting that there is no Infl under considers.

The current analysis of this is that Bill is in this case in the specifier of AP, serving as “the subject of the adjective phrase”

(13) \[
\begin{array}{c}
\text{AP} \\
\text{Spec} \\
\text{NP} \\
\text{Bill} \\
\text{incompetent}
\end{array}
]\]

This seems to be possible with other phrases as well—

(14) a. The captain expects the drunken sailor off the ship (immediately).  
    b. John made Bill read the whole book.

(15) \[
\begin{array}{c}
\text{PP} \\
\text{Spec} \\
\text{NP} \\
\text{the drunken sailors} \\
\text{P} \\
\text{off} \\
\text{the ship}
\end{array}
]\]
So, phrases of all kinds seem to be able to have subjects in their specifier, under certain conditions anyway. Notice that these small clauses seem to have no independent tense interpretation—tense is a property of Infl, and where there is no Infl, there is no tense.

Subjects, IP, and the Extended Projection Principle

Last time, we talked about CP in questions, which has a [+Q] C. We observed that it seems to be true that a [+Q] C requires a specifier (with a [+wh] phrase in it).

Like CP in questions, it turns out that IP seems to require a specifier (unlike most of the other categories). This is encoded as follows:

**Extended Projection Principle (EPP)**

Clauses must have a subject (that is, ‘The specifier of IP must be filled.’).

(The name really has very little to do with the conceptual content—you might be better off just forgetting about the words and refer to this as the “EPP”).

One bit of evidence comes from the raising cases; where we can’t raise the embedded subject (for example, when it violates the TSC), we have to put a meaningless element (expletive) *it* in subject position.

(17)  
   a. *(It) seems that Mary has solved the problem.
   b. Mary seems *it* to have solved the problem.

A type of sentence of this sort which has gotten a great deal of attention are *there*-constructions like:

(18)  
   a. A unicorn is in the garden.
   b. *(There) is a unicorn in the garden.

In (18b), the meaningless element *there* is required. What makes it required is the EPP.
Incidentally: Where is a *unicorn* in (18a)? A good guess, given the small clauses discussed earlier, is that it is the subject of a PP:

(19)

```
           IP
           \   /  \
Spec   I'
       \   /  \
        I  VP
          \ /  \
          [pres] V'
               \   /  \
               V    PP
                 \ /  \
                 be PP
                   \ /  \
                   NP    P'
                     \ /  \
                     a unicorn    NP
                       \ /    \    /  \
                      P   in   the garden
```

And, for (18b), perhaps *there* fills SpecIP, satisfying the EPP, which allows a *unicorn* to stay in SpecPP.

One other thing: in (20), there are two IPs. Where is the subject of the lower one?

(20) 

```
[IP Mary_i seems [IP t_i to have solved the problem]].
```

*Answer:* The trace counts—further evidence that we need traces. SpecIP is filled, but with the *trace* of Mary. The EPP is satisfied.

---

**Nonfinite clauses and PRO**

(21) a. John tried [CP [IP e to leave]].
   b. John persuaded Bill [CP [IP e to leave]].
   c. It is difficult [CP [IP e to leave]].

Here there appears to be no subject (*EPP*) in the lower (nonfinite) clause. The subject is not a trace. *Try, persuade, be difficult* are not raising verbs (like *seem*).

(22) a. *There tried John to be a unicorn.*
   b. *It seems John to leave.*

The embedded agent (the one leaving) is *John* in (21a), **but** it is *Bill* in (21b), and *someone/anyone* in (21c).
So: • There must be something there (by the EPP).
• We can’t hear it.
• It can’t be a trace of movement.

→ It must be an unpronounced element (and like a pronoun).

(23) a. Johni tried \([\text{CP} \ [\text{IP} \ \text{PRO}_i \ \text{to leave}]]\).  
    \textit{Subject control}

b. John persuaded Billi \([\text{CP} \ [\text{IP} \ \text{PRO}_i \ \text{to leave}]]\).  
    \textit{Object control}

c. It is difficult \([\text{CP} \ [\text{IP} \ \text{PRO}_{arb} \ \text{to leave}]]\).  
    \textit{Arbitrary control}

PRO only appears in nonfinite clauses.
Nonfinite clauses lack tense and agreement features in Infl:

(24) \[
\begin{array}{c}
\text{Spec} \\
\text{I'} \\
\text{I} \\
\text{VP} \\
\text{(to)} \\
\text{…}
\end{array}
\]

Not all nonfinite clauses have to—gerundive clauses are nonfinite but lack to:

(25) a. John dislikes \([\text{CP} \ [\text{IP} \ \text{PRO} \ \text{eating in public}]]\).

b. \([\text{CP} \ [\text{IP} \ \text{PRO} \ \text{reading detective stories}]]\) is fun.

\begin{tcolorbox}
\textbf{Chapter 7: $\theta$-theory}
\end{tcolorbox}

The structure of the framework we are developing is something like this:

\[
\begin{array}{c}
\text{X’ Theory} \\
\text{Case Theory} \\
\text{Binding Theory}
\end{array}
\]

\[
\begin{array}{c}
\text{\theta-Theory}
\end{array}
\]

Where there are several “modules” that each make decisions about which structures are well-formed and which are not. We’ve seen the restrictions that X’-theory places on structures, and now we will turn to restrictions that \theta-Theory places on structures. (Case Theory and Binding Theory are coming up in the future).

Different verbs have different kinds (and numbers) of \textit{arguments} that contribute to the meaning of the sentence. Differences of this kind are the kind we put in the lexical entries of the verb.
The goalie kicked the ball.
Pat likes pizza.
Joe gave food to the cat.

The goalie is the agent initiating the kicking, the ball is the thing kicked. Pat is the one experiencing liking (not really an agent initiating a liking), pizza is the thing liked.

So:
The subject of kick has the \(\theta\)-role of Agent.
The object of kick has the \(\theta\)-role of Patient.
The subject of like has the \(\theta\)-role of Experiencer.
The object of like has the \(\theta\)-role of Theme.
The recipient of give has the \(\theta\)-role of Goal.

The \(\theta\)-role is the semantic role (thematic role) played by the argument in the event.

An argument is a ‘referring expression’—corresponds to an individual or entity (perhaps abstract). Verbs select for arguments on the basis of their syntactic category (c-selection), and on the basis of their semantic properties (s-selection).

A verb in the lexicon comes with a list of the \(\theta\)-roles its arguments will play.

(27) 
\[
\begin{array}{l}
\text{kick: Agent, Patient} \\
\text{like: Experiencer, Theme}
\end{array}
\]

Important concept: The verb is thought of as assigning roles to its syntactic arguments.

(28) The goalie kicked the ball.
\[
\begin{array}{c}
\text{Agent} \\
\text{Patient}
\end{array}
\]

So, a verb has an Agent slot and a Patient slot, and it finds arguments in the structure to assign its \(\theta\)-roles to.

Internal and external arguments:

Simply knowing that \textit{kick} has an Agent and a Patient doesn’t tell us which one is the subject and which is the Object. We need to say something about which \(\theta\)-role gets assigned to which structural position. We divide the \(\theta\)-roles into \textbf{internal \(\theta\)-role} and \textbf{external \(\theta\)-role} based on the structural position the arguments are found:
Verbs directly θ-mark the internal argument, and indirectly θ-mark the external argument.

θ-positions vs. θ′-positions.

So, there are certain places in the structure that receive θ-roles: sister of V, SpecIP. These structural positions are called θ-positions. Positions where no θ-role is assigned are called θ′-positions.

In many sentences, SpecIP is a θ-position, where, e.g., the Agent θ-role is assigned.

There are verbs which do not assign an external θ-role. For example, raising verbs like seem:

(31) a. It seems [CP that [IP Mary has solved the problem]].
   b. Mary, it seems [IP t i to have solved the problem]].

In (31a), (expletive) it is not an argument (it is not a referring expression). It appears that seem should be represented like:

(32) seem: Ø <proposition>

…meaning that the root clause SpecIP in (31a) is not a position which receives a θ-role. It is a θ′-position.

In (31b), Mary occupies the root clause SpecIP (having been moved from the embedded clause), but it is still occupying a θ′-position—The θ-role Mary receives comes from the embedded verb solve.

The θ-criterion (first statement)

i) Each argument must be assigned a θ-role.

ii) Each θ-role must be assigned to an argument.
It seems sensible, and it also seems right…

(33) a. * John seems that Mary has solved the problem.
    b. * There solved a problem.
    c. * Mary solved there.

Now, wait, consider (31b). Mary is an argument in a $\theta'$-position—how does Mary get its $\theta$-role from *solve* way up there?

**Chains:** The collection of positions occupied by a single argument (its base position and its derived positions).

(34) \[
\begin{array}{c}
\text{Mary}_i \\
\text{seems } \{ \text{IP } t_i \} \\
to \text{have solved the problem}]].
\end{array}
\]

\[
\text{Chain: } \{ \text{Mary}_i, t_i \}
\]

If we think of $\theta$-roles as assigned to words (specifically, the *bottom* of chains), we solve the terminological problem.

**The $\theta$-criterion**

i) Each argument chain must be assigned exactly one a $\theta$-role.

ii) Each $\theta$-role must be assigned to exactly one argument chain.

Requires that we consider unmoved arguments to be **trivial chains**.

Note: This means that you can never move an argument into a $\theta$-position. Arguments can only move from a $\theta$-position to a $\theta'$-position.

Arguments as referring expressions and covert *wh*-movement.

**Arguments**, recall, are referring expressions.

But there are plenty of things which appear to sit in argument position that are not referring expressions.

(35) a. Which problem did Mary solve?
    b. DS: \[
    [\text{CP } e \: \text{[+Q]} \: \text{[IP } \text{Mary } [\text{r } \text{Tense } [\text{vp } \text{solve which problem}]]]]
    \]
    c. SS: \[
    [\text{CP } \text{which problem}_i \: \text{did}_j \: \text{[IP } \text{Mary } [\text{r } t_j [\text{vp } \text{solve } t_i ]]]]
    \]

*solve* assigns a $\theta$-role to the object position, but there is no argument there—*which problem* is not a referring expression.

However, *traces* in these situation have the status of **logical variables**, and will thereby count as referring expressions. So, the $\theta$-role can be assigned by *solve* to the *trace* of *which problem*, even though it couldn’t be assigned to *which problem* itself.
Here’s a peek at how we interpret this question:

(36) for which problem \( x \): [Mary solved \( x \)].

Suppose \( x \) is problem #3. That is a referring expression. The idea is: Whatever you choose for \( x \), it will be a referring expression, making the trace (interpreted as \( x \)) eligible to receive a \( \theta \)-role.

So, \( wh \)-movement is forced in order to satisfy the \( \theta \)-criterion. The \( \theta \)-criterion is not satisfied at DS, but it is satisfied at SS.

But consider multiple questions:

(37) a. Who solved which problem?
b. What did John give to whom?

If solve needs to assign a \( \theta \)-role to the object and give needs to assign a \( \theta \)-role to the indirect object, it can’t do that at SS, since only one \( wh \)-phrase moves to SpecCP.

If everything we’ve said so far is right, we need to move both \( wh \)-phrases in order to satisfy the \( \theta \)-criterion; it needs to be interpreted like:

(38) for which person \( x \) and which problem \( y \), [\( x \) solved \( y \)].

That is, we need to do covert \( wh \)-movement.

The structure of the grammar:

<table>
<thead>
<tr>
<th>The structure of the grammar:</th>
<th>DS ↙ phrase structure rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>surface structure (abstract)</td>
<td>SS ↙ movement rules</td>
</tr>
<tr>
<td>“phonetic form”</td>
<td>PF ↙ more movement rules</td>
</tr>
<tr>
<td></td>
<td>LF ↙ “logical form” (meaning)</td>
</tr>
</tbody>
</table>

This implies a couple of things:

- The \( \theta \)-criterion applies at LF.
- \( wh \)-movement after SS (covert \( wh \)-movement) adjoins to SpecCP.

SpecCP is already filled by who, the first \( wh \)-phrase moved before SS.

(39)   
\[
\begin{array}{c}
\text{NP} \\
\text{CP} \\
\text{NP} \\
\text{NP} \\
\text{C} \\
\text{IP} \\
\text{who} \\
\text{which prob} \\
\text{[+Q]} \\
\text{…}
\end{array}
\]
Incidentally, this is not crazy. To take a very brief excursion away from English, consider the following, from Bulgarian. In Bulgarian, all wh-phrases in a multiple question are moved to the front of a sentence—arguably exactly the structure in (39), assuming yesterday is adjoined to IP.

(40)  a. *koj kogo vchera e udaril*  
     who whom yesterday has hit  
     ‘Who hit whom yesterday?’

    b. ?? *koj vchera kogo e udaril*  
     who yesterday whom has hit  
     (‘Who hit whom yesterday?’)

So, we would say that Bulgarian (and other languages which do this multiple wh-fronting) does all of its wh-movement overtly (by SS) whereas English moves only one wh-phrase overtly, and does the rest of its wh-movement covertly (after SS).

Quantifiers, QR, and c-command

Quantifiers like *everyone, someone, noone* are also not referring expressions. Just like wh-phrases, they need to move in order to satisfy the θ-criterion.

(41)  a. John suspects everyone.
   b. SS: [IP John suspects everyone].
   c. LF: [IP everyone, [IP John suspects ti]].

(42) 

(43) For every person x: [John suspects x].
Everyone suspects someone.

a. For every person x [ there is a person y [ x suspects y ]].
   ‘For everyone x, you can find a person y such that x suspects y.’

b. There is a person y [ for every person x [ x suspects y ]].
   ‘There is a person y such that y is suspected by everyone.’

Two logical scope readings—we match the logical structure to syntactic structure:

We can define a notion of scope that predicts the readings:

**Scope**
The scope of $\alpha$ is the set of nodes $\alpha$ c-commands in the LF representation.

**C-command**
a c-commands b iff:
   i) the first branching node dominating $\alpha$ also dominates $\beta$.
   ii) $\alpha$ does not dominate $\beta$. 

C-command is actually a pretty easy concept, but it’s clearer from a picture.

\[(49)\]

\[
\begin{array}{c}
A \\
B \\
C \\
D \\
E
\end{array}
\]

B c-commands C, D, and E
D c-commands E (and vice versa)
C c-commands B (and vice versa)

Informally: To find what a node c-commands, go up one level, and it is everything below it except the original node.

\[(50)\]

\[
\begin{array}{c}
IP_1 \\
NP_j \\
someone \\
NP_i \\
everyone \\
NP_i \\
IP_3 \\
\ldots
\end{array}
\]

In (50), someone c-commands everyone, and in (51) everyone c-commands someone.

So in (50), someone has scope over everyone,
and in (51) everyone has scope over someone.

For this to work formally requires introducing some terminology about adjunction—

\[(52)\]

\[
\begin{array}{c}
XP_1 \\
adjunct \\
XP_2 \\
adjunct \\
XP_3 \\
\ldots
\end{array}
\]

category (XP)

segments (XP_1)

segments (XP_2)

segments (XP_3)

Segments count for determining c-command.
A-positions vs. A′-positions; A-chains vs. A′-chains

An **A-position** (argument position) is a structural position where an argument can be found at LF. For example, subject position (SpecIP), object position (complement of V).

An **A′-position** is a structural position where a non-argument can be found at LF.

We’ve been talking about quantifiers and wh-phrases, which are non-arguments by virtue of their not being referring expressions. We find them in SpecCP and adjoined to IP. Accordingly, SpecCP and adjoined positions are A′-positions.

Recall the: θ-position vs. θ′-position difference from before (A θ-position being a position where an argument gets a θ-role). Obviously, any A′-position is also a θ′-position (if you have a non-argument there, it can’t be getting a θ-role there). But there are A-positions which are not θ-positions. SpecIP is a θ-position only when the verb assigns an external argument, but it is always an A-position.

It is also useful to distinguish movement chains into **A-chains** (movement to an A-position) and **A′-chains** (movement to an A′-position).

Nonstandard arguments

Passives

(53)  a. Mary ate the sandwich.
     b. The sandwich was eaten.

The passive verb does not assign an external θ-role (to SpecIP).
We create the passive verb by attaching -ed or -en to the verb (remains a verb).

We consider this derivation to take place *in the lexicon* (prior to insertion into X′ trees). Attaching -en suppresses the external θ-role:

(54)   

<table>
<thead>
<tr>
<th>agent</th>
<th>theme</th>
</tr>
</thead>
<tbody>
<tr>
<td>eat:</td>
<td>Agent</td>
</tr>
<tr>
<td>eaten:</td>
<td>—</td>
</tr>
</tbody>
</table>
SpecIP is an A-position.
SpecIP is a θ'-position.
{the sandwich, t_i} is an A-chain.

EPP (SpecIP must be filled) forces movement of the sandwich to SpecIP.

Unaccusatives

There is a class of intransitive verbs that work in a very similar way to the passive.

(56)  a. The vase broke.
     b. John broke the vase.

The standard analysis of this type of intransitive verb is that they have an internal argument but no external argument.

(57)  break: Ø <Theme>

(58)  [IP [the vase], [VP broke t_i]]

As for (56a), it means something like ‘John caused the vase to break’ and is usually analyzed as having a “causative” morpheme that adds an external Agent argument (sort of the opposite of the passive morpheme, which removes the external argument).

(59)  [IP John [VP CAUSE+break the vase]]

(60)  CAUSE+break: Agent <Theme>
Adjectival passives

The adjectival passives are like the verbal passives (active object appears in subject position) but are adjectives (as we can see by the ability to prefix un- and the fact that they can modify nouns, selected by seem and remain).

(61)  
  a. The island was uninhabited.
  b. The performance was interrupted.

(62)  
  a. CBS employees inhabited the island.
  b. John interrupted the performance.

(63)  
  a. The uninhabited island
  b. The uninterrupted performance

(64)  
  a. The island seemed uninhabited.
  b. The performance remained uninterrupted.

Derived in the lexicon with a category changing suffix (also -en, -ed):

(65)  
  inhabit: [V -N] \rightarrow inhabited: [V +N]
  Agent <Theme> \rightarrow \emptyset <Theme>