1. Agreement

**Agreement**: when constraints hold among constituents that take part in a rule or set of rules

For example, in English, as in many other languages, determiners and the head nouns in NPs have to agree in number

- `this flight`  
- `this flights`  
- `*those flight`  
- `those flights`

2. The agreement problem for CFGs

Our earlier NP rules are clearly deficient since they don’t capture this constraint

```
NP → Det Nominal
```

- That rule accepts, and assigns correct structures, to grammatical examples (`this flight`)
- But also accepts incorrect examples (`*these flight`)

Such a rule is said to **overgenerate**

- *We’ll come back to this...*

3. Verb Phrases

English verb phrases consist of

- some optional modifiers
• a main verb
  ◦ which we will once again call the **head**
• and zero or more **arguments**
  ◦ also called **complements**

### 4. Verb Phrases: modifiers

We have to account for a range of structures ahead of the main verb

- Including adverbs, modals and auxiliary verbs

```
leave  Verbal → V
may leave  Verbal → Modal Verbal
has left  Verbal → Aux Verbal
suddenly left  Verbal → Adv Verbal
```

We get a familiar-looking right-branching structure when these combine

![Diagram of verb phrase structure](image)

### 5. Subcategorisation

We need some rules for different patterns of arguments:

**intransitive**

```
disappear  VP → Verbal
```

**transitive**

```
preferring a morning flight  VP → Verbal NP
```

**transitive with PP**

```
leave Boston in the morning  VP → Verbal NP PP
```
ditransitive

buy Robin a ticket

Not all verbs are allowed to participate in all the VP rules

We **subcategorise** verbs in a language according to the sets of VP rules they participate in

This is a modern take on the traditional notion of transitive/intransitive.

Modern grammars may have 100s of subcategorisation classes

- Sometimes called subcategorisation **frames**

### 6. Subcat examples and counterexamples

Some examples of the diversity of complement patterning

<table>
<thead>
<tr>
<th>John sneezed</th>
<th>Please find a flight to Edinburgh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can you help me with a flight</td>
<td>Give me a cheaper fare</td>
</tr>
<tr>
<td>Give a cheaper fare to my children</td>
<td>I prefer to leave earlier</td>
</tr>
<tr>
<td>I was told (that) KLM has a flight</td>
<td></td>
</tr>
</tbody>
</table>

And some counterexamples

*John sneezed the book
*I prefer KLM has a flight
*Give with a flight

As with agreement phenomena, we need a way to formally express the constraints

### 7. Overly complicated, and wrong as well?

(Before we go on to agreement, a brief diversion)

You might feel that all these (mostly binary) rules are missing the point

- Particularly, because they allow all kinds of wrong orders

Why don’t we just make the order explicit?

 struggles
That is, why not allow regular expressions over $T \cup NT$ on the right hand side?

- We could, and people have
- Either as an extension to CFGs
- Or as an extension to FSAs, called **Pushdown Automata**
  - Or sometimes **Recursive Transition Networks**

### 8. Extending CFGs

You can understand such an extension to CFGs in one of two ways:

- As a change to the formalism itself, i.e.
  - $\text{rhs}$ a regular expression whose alphabet is $T \cup NT$
  - corresponding (non-trivial) changes to the rewriting and node-admissibility definitions
- As an extension to the notation *only*, not to the formalism as such
  - i.e., we treat rules notated like so:

  \[
  X \rightarrow \ldots \ Y \ldots
  \]

  - As just shorthand notation for the more verbose pair of notations
  \[
  X \rightarrow \ldots \ Y \ldots \\
  X \rightarrow \ldots \ Y \ldots
  \]

On this account, our VP 'rule' on the previous slide is a shorthand notation for *eight* actual rules.

What about the NP rule, with its Kleene star?

### 9. Infinite CFGs

Including Kleene star in our notation for the right-hand side of rules turns out to have a surprising consequence

If we take the same approach as we did for question-mark

- i.e., we treat rules notated like so:

  \[
  X \rightarrow \ldots \ Y \ldots
  \]
• As just shorthand notation for the more verbose pair of notations
  \[ X \rightarrow \ldots_1 \ldots_2 \]
  \[ X \rightarrow \ldots_1 Y Y^* \ldots_2 \]

Either way, we have what amounts to (a notation for) a CFG with an *infinite* number of rules!

• That actually has the potential to change the status of the formalism
  ◦ Its *weak generative capacity*
  ◦ AKA its position on the *Chomsky hierarchy*

[End of diversion]

10. Why constraints?

The NP and VP rules we’ve seen so far *overgenerate*

• They permit the presence of strings containing
  ◦ Determiners and nouns that don’t go together
  ◦ Verbs and arguments that don’t go together

11. Possible CFG Solution for Agreement

We could try to address our agreement problems by expanding the non-terminal categories to encode agreement:

\[
\begin{align*}
& \text{NP}_{sg} \rightarrow \text{CNP}_{sg} \\
& \text{CNP}_{sg} \rightarrow \text{Det}_{sg} \text{CNP}_{sg} \\
& \text{NP}_{pl} \rightarrow \text{CNP}_{pl} \\
& \text{CNP}_{pl} \rightarrow \text{Det}_{pl} \text{CNP}_{pl} \\
& S_{sg} \rightarrow \text{NP}_{sg} \text{VP}_{sg} \\
& S_{pl} \rightarrow \text{NP}_{pl} \text{VP}_{pl} \\
& \text{VP}_{pl} \rightarrow V_{pl} \text{NP} \\
& \text{VP}_{sg} \rightarrow V_{sg} \text{NP}
\end{align*}
\]

And a *lot* more besides

Can use the same trick for all the verb/VP classes

• But this clearly has become quite obscure
  • And the (multiplicative) interaction *between* number agreement and subcategorisation will make things *much* worse

12. CFG Solution for Agreement

**Good thing**

It works and stays within the power of CFGs
Less good things
  • It's inelegant
  • It doesn't scale
    ◦ The interaction among various families of constraints explodes the number of categories and rules in the grammar
  • It still overgenerates!
    ◦ It can't deal with unbounded dependency

13. CFG conclusions

CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English

But there are problems
  • Agreement
  • Unbounded dependencies

There are more elegant solutions
  • But they go beyond the formal power of CFGs
    ◦ Sign-based theories (GPSG, HPSG)
    ◦ Tree-adjoining grammars

We'll look at LTAG, one variety of tree-adjoining grammar, next week

But first, we'll expand our approach to categories
  • By adding features

14. Features

The name feature has been around in Linguistics for a long time

The basic idea is to capture generalisations by decomposing monolithic categories into collections of simpler features

Originally developed for phonology, where we might have e.g.

/ɪ/  +high, +front
/e/  -high, +front
/o/  -high, -front
/u/  +high, -front

Where we can now 'explain' why /ɪ/ and /u/ behave similarly in certain cases, while /i/ and /e/ go together in other cases.

Those are all binary features
  • sometimes also used at the level of syntax:
    • +/- singular; +/- finite
15. Features, cont'd

But more often we find features whose values are some enumerated type

| person: {1st, 2nd, 3rd}; number: {sg, pl}; ntype: {count, mass} |

We'll follow J&M and write collections of features like this:

| person 3rd |
| number pl |

It will be convenient to generalise and allow features to take feature bundles as values:

| ntype count |
| agreement person 3rd |
| agreement number pl |

16. Features in use

We can now add feature bundles to categories in our grammars

In practice we allow some further notational conveniences:

- Not all features need be specified (e.g. the number feature for 'sheep')
- In rules, we allow the values of features to be variables
- And we can add constraints in terms of those variables to rules

For example

| ntype count |
| agreement person 3rd |
| agreement number pl |

\[ N[ \text{agreement } x ] \rightarrow \text{D[ agreement } y ] N[ \text{agreement } z ] \ x = y = z \]