

# Syntactic Features

Morphology, heads, gaps, etc.

# 3 views of a context-free rule

- generation (production):  $S \rightarrow NP VP$
- parsing (comprehension):  $S \leftarrow NP VP$
- verification (checking):  $S = NP VP$
- Today you should keep the third, declarative perspective in mind.

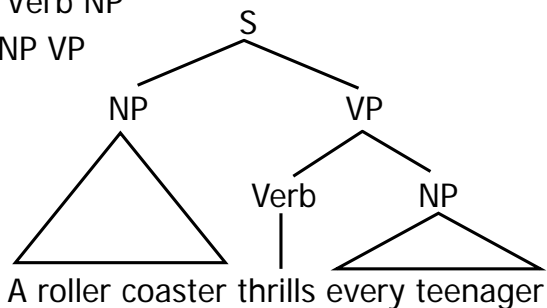
- Each phrase has
  - an interface (S) saying where it can go
  - an implementation (NP VP) saying what's in it
- To let the parts of the tree coordinate more closely with one another, enrich the interfaces:
 
$$S[\text{features...}] = NP[\text{features...}] VP[\text{features...}]$$

# Examples

Verb  $\rightarrow$  thrills

VP  $\rightarrow$  Verb NP

S  $\rightarrow$  NP VP



# 3 common ways to use features

morphology of a single word:

Verb[head=thrill, tense=present, num=sing, person=3,...]  $\rightarrow$  thrills

projection of features up to a bigger phrase

VP[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...]  $\rightarrow$  V[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...] NP  
provided  $\alpha$  is in the set TRANSITIVE-VERBS

agreement between sister phrases:

S[head= $\alpha$ , tense= $\beta$ ]  $\rightarrow$  NP[num= $\gamma$ ,...] VP[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...]

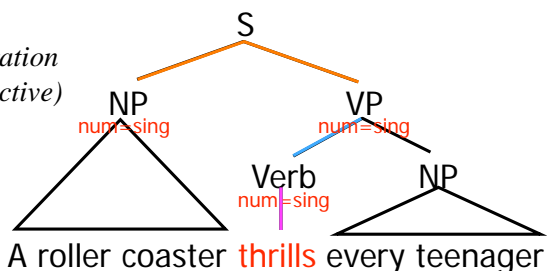
# 3 Common Ways to Use Features

Verb[head=thrill, tense=present, num=sing, person=3,...]  $\rightarrow$  thrills

VP[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...]  $\rightarrow$  V[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...] NP

S[head= $\alpha$ , tense= $\beta$ ]  $\rightarrow$  NP[num= $\gamma$ ,...] VP[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...]

(generation perspective)



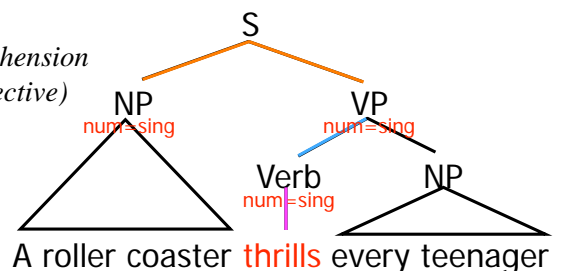
# 3 Common Ways to Use Features

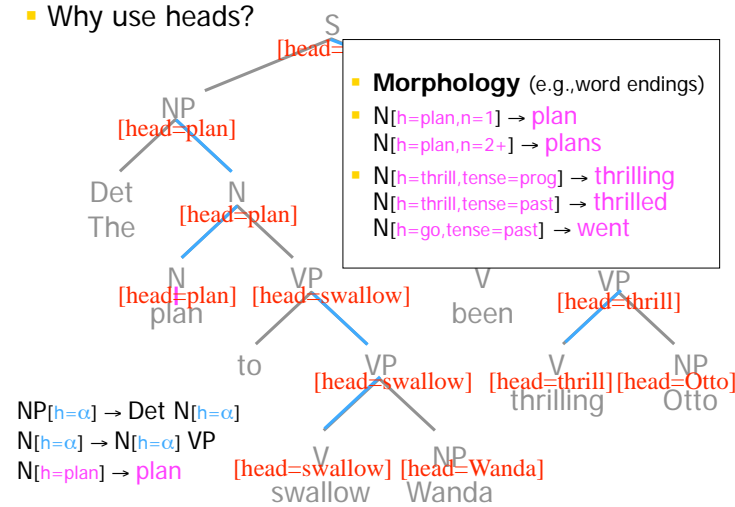
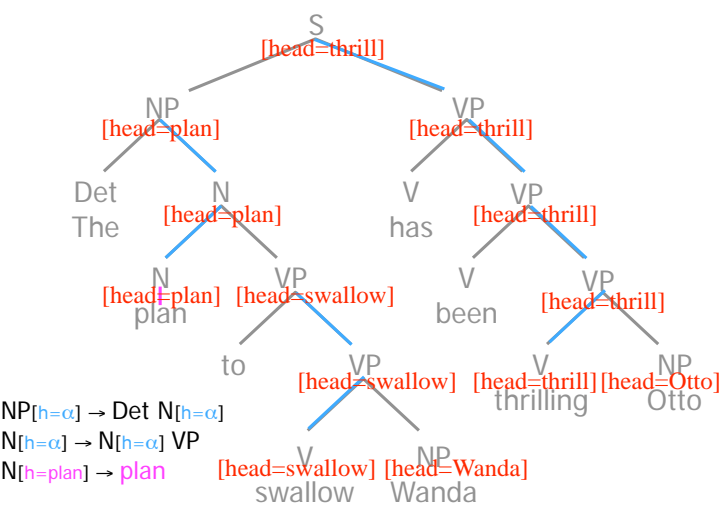
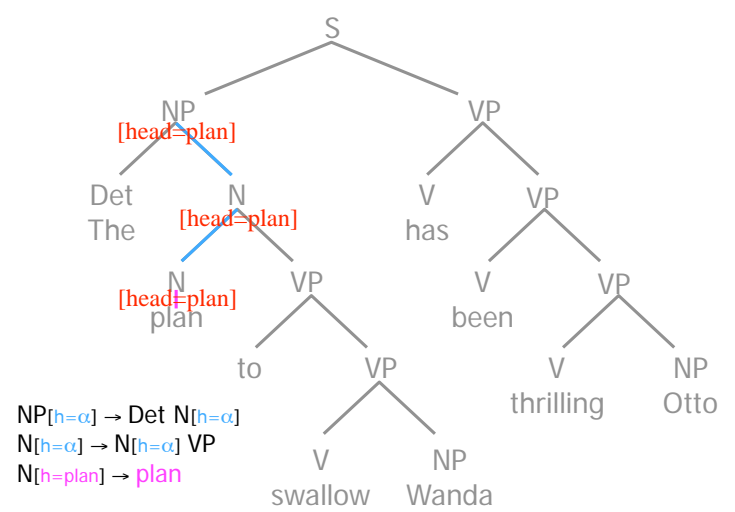
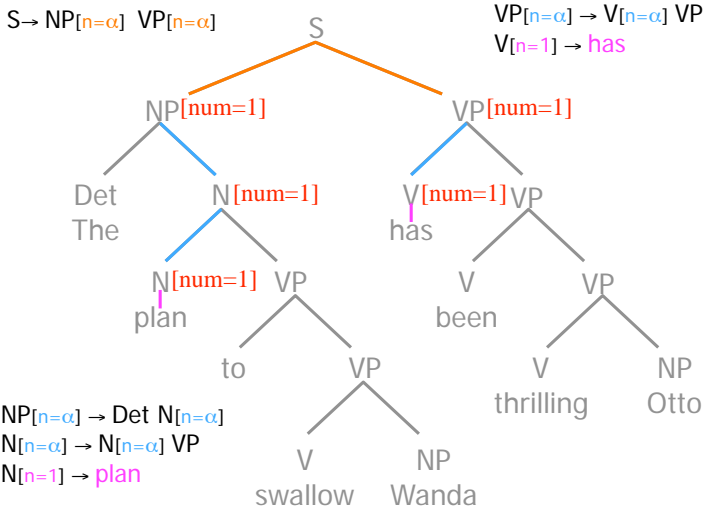
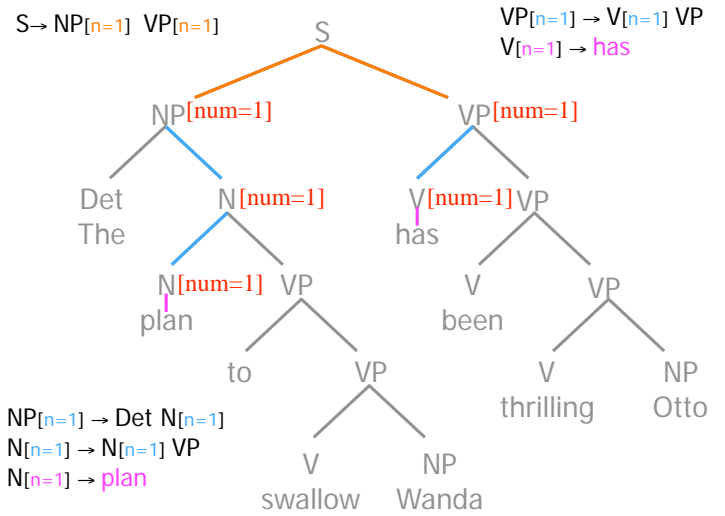
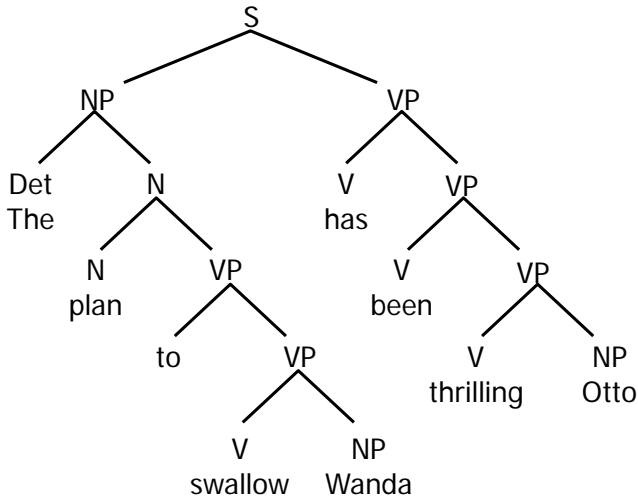
Verb[head=thrill, tense=present, num=sing, person=3,...]  $\rightarrow$  thrills

VP[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...]  $\rightarrow$  V[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...] NP

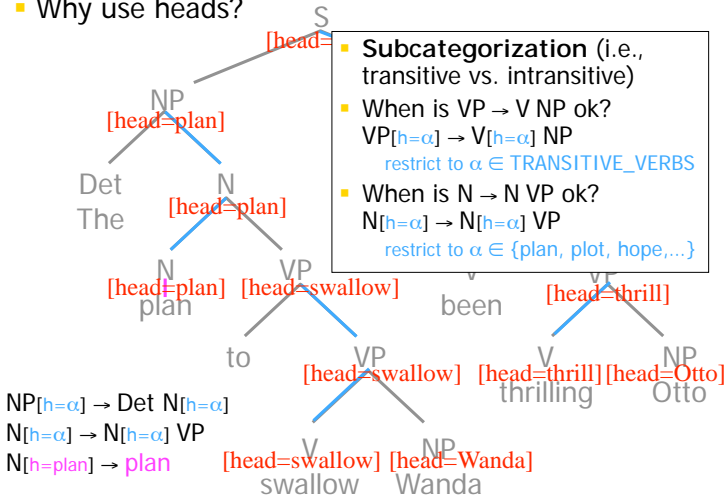
S[head= $\alpha$ , tense= $\beta$ ]  $\rightarrow$  NP[num= $\gamma$ ,...] VP[head= $\alpha$ , tense= $\beta$ , num= $\gamma$ ...]

(comprehension perspective)

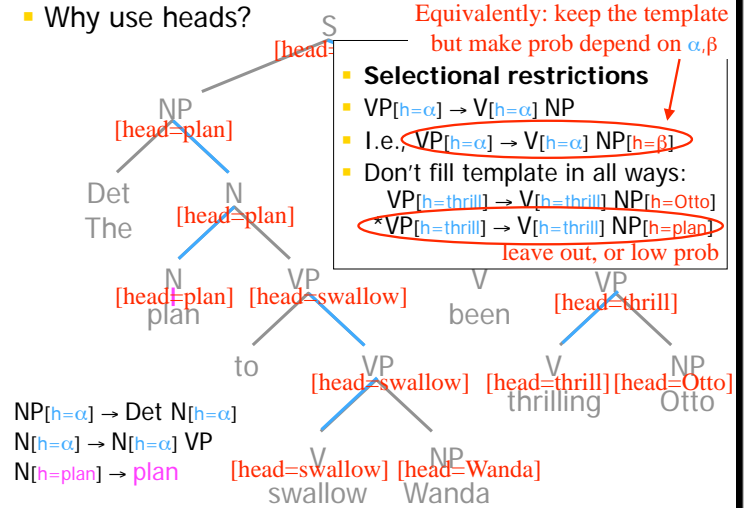




Why use heads?

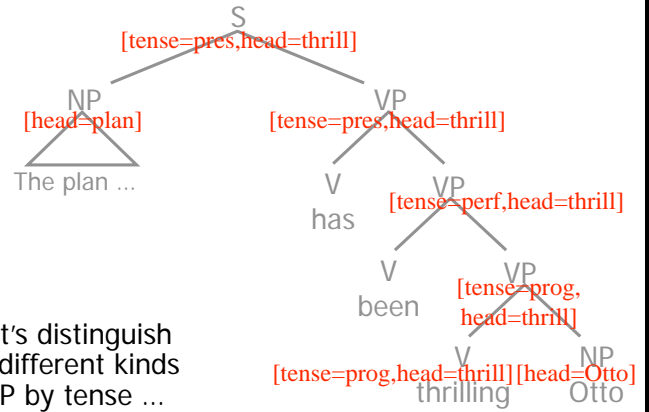


Why use heads?

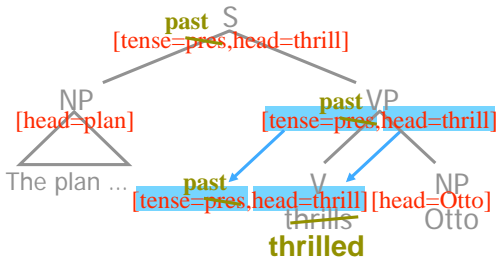


Part of the English Tense System

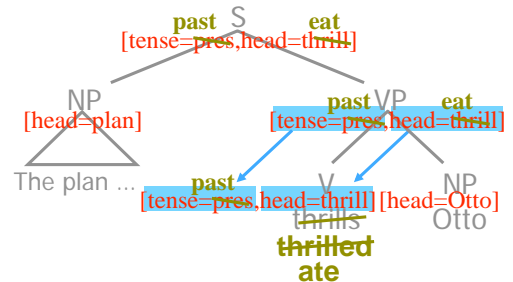
	Present	Past	Future	Infinitive
Simple	eats	ate	will eat	to eat
Perfect	has eaten	had eaten	will have eaten	to have eaten
progressive	is eating	was eating	will be eating	to be eating
Perfect+ progressive	has been eating	had been eating	will have been eating	to have been eating



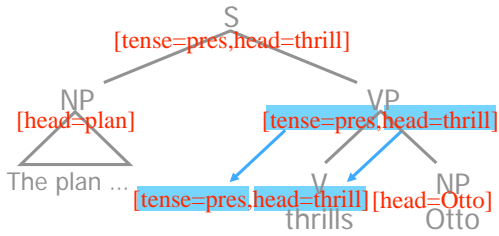
Let's distinguish the different kinds of VP by tense ...



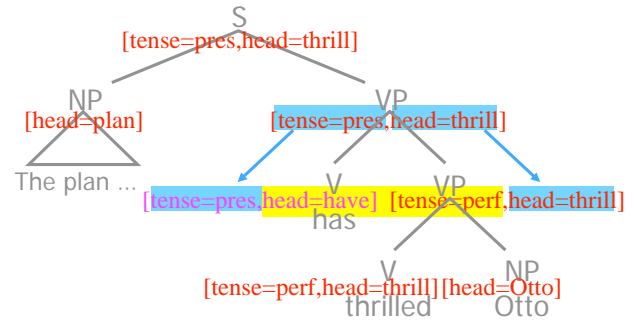
**Past**  
 Present tense



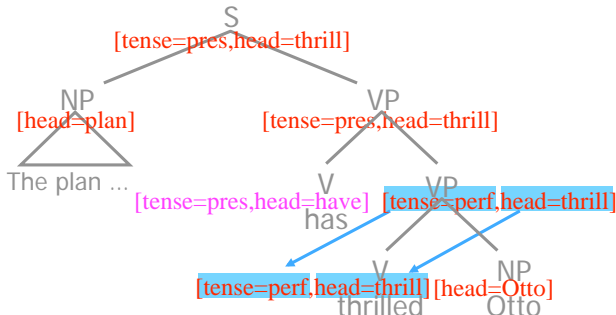
**Past**  
 Present tense



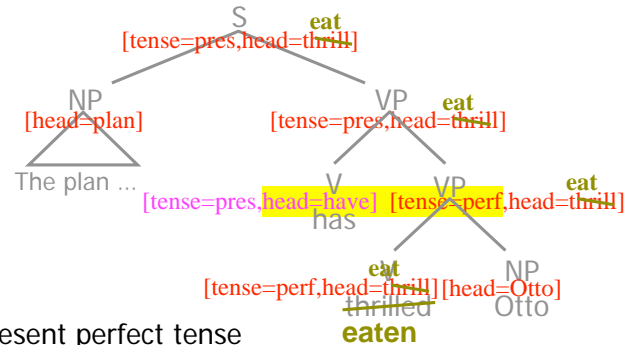
- Present tense (again)



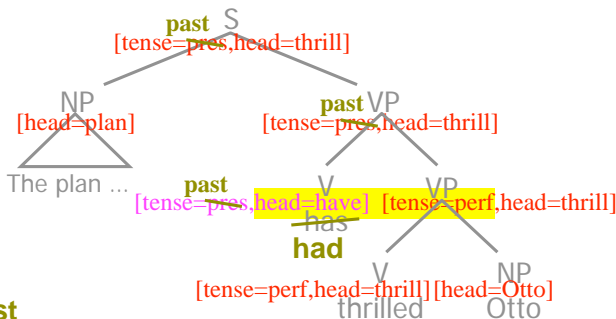
- Present perfect tense



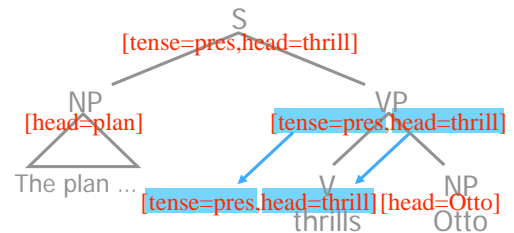
- Present perfect tense



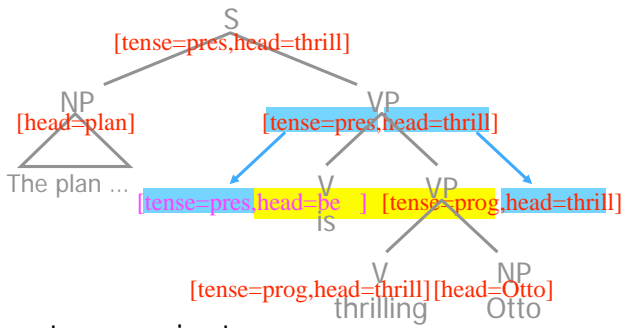
- Present perfect tense
- The yellow material makes it **not ate** – why? a perfect tense – what effects?



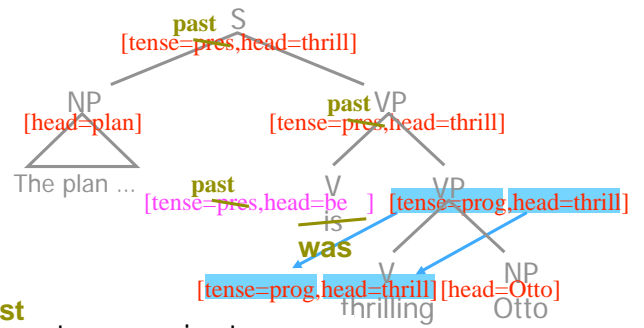
- Past** Present perfect tense



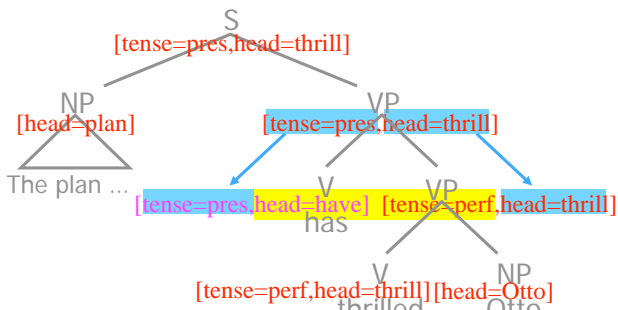
- Present tense (again)



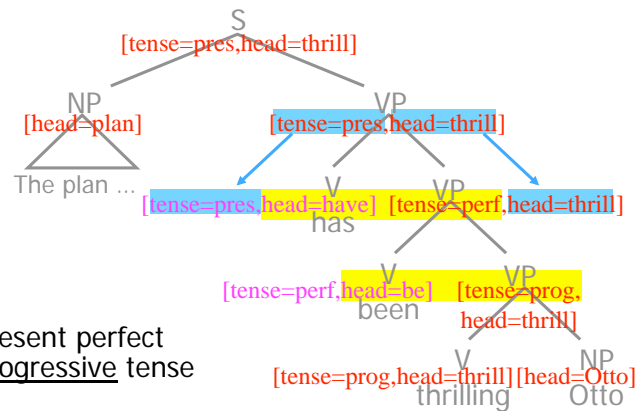
- Present progressive tense



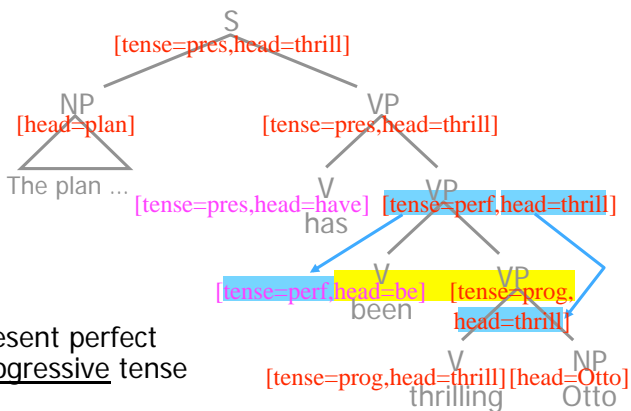
- Past**
- Present progressive tense



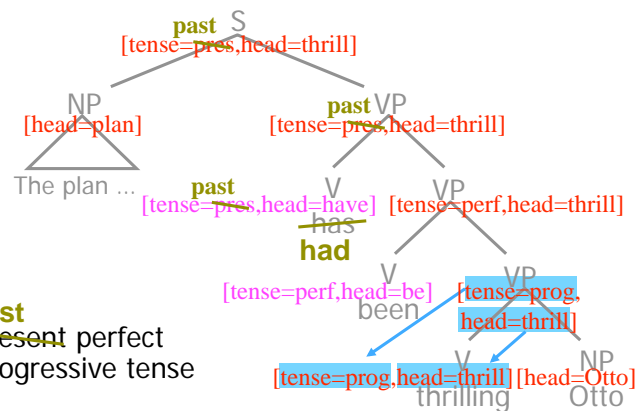
- Present perfect tense (again)



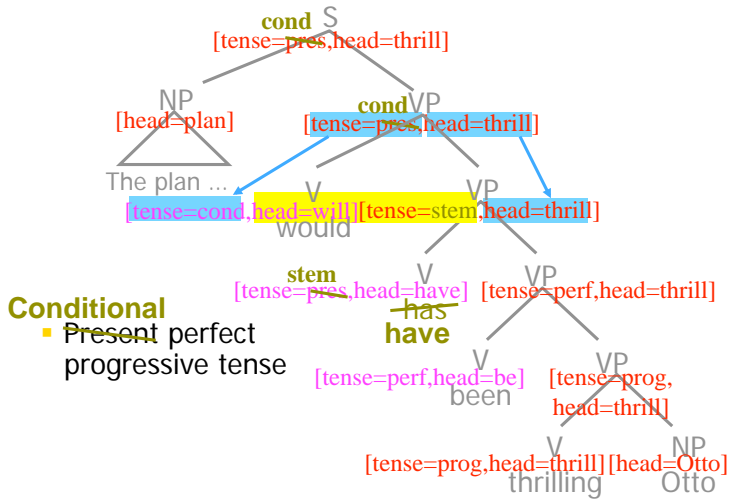
- Present perfect progressive tense



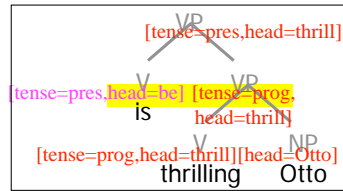
- Present perfect progressive tense



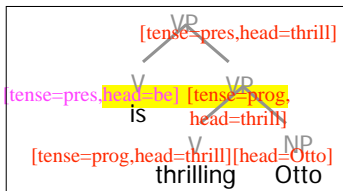
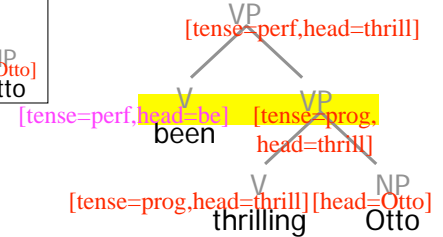
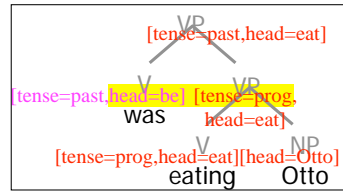
- Past**
- Present perfect progressive tense



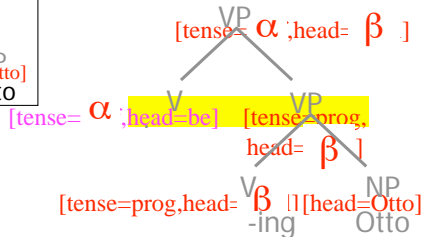
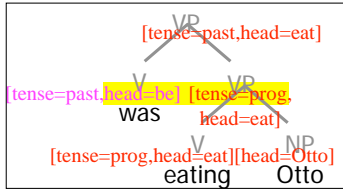
**Conditional**  
 Present perfect progressive tense



So what pattern do all progressives follow?



So what pattern do all progressives follow?



- Progressive: VP[tense=α, head=β, ...] → V[tense=α, stem=be ...] VP[tense=prog, head=β ...]
- Perfect: VP[tense=α, head=β, ...] → V[tense=α, stem=have ...] VP[tense=perf, head=β ...]
- Future or conditional: VP[tense=α, head=β, ...] → V[tense=α, stem=will ...] VP[tense=stem, head=β ...]
- Infinitive: VP[tense=inf, head=β, ...] → to VP[tense=stem, head=β ...]
- Etc. VP[tense=α, head=β ...]

As well as the "ordinary" rules:  
 VP[tense=α, head=β, ...] → V[tense=α, head=β, ...] NP [tense=prog, head=β ...] [head=Otto]  
 V[tense=past, head=have ...] → had

## Gaps ("deep" grammar!)

- Pretend "kiss" is a pure transitive verb.
- Is "the president kissed" grammatical?
  - If so, what type of phrase is it?
- the sandwich that the president kissed e
- I wonder what Sally said the president kissed e
- What else has Sally consumed the pickle with e

## Gaps

- **Object gaps:**
- the sandwich that the president kissed e
- I wonder what Sally said the president kissed e
- What else has Sally consumed the pickle with e
- What else has Sally consumed e with the pickle

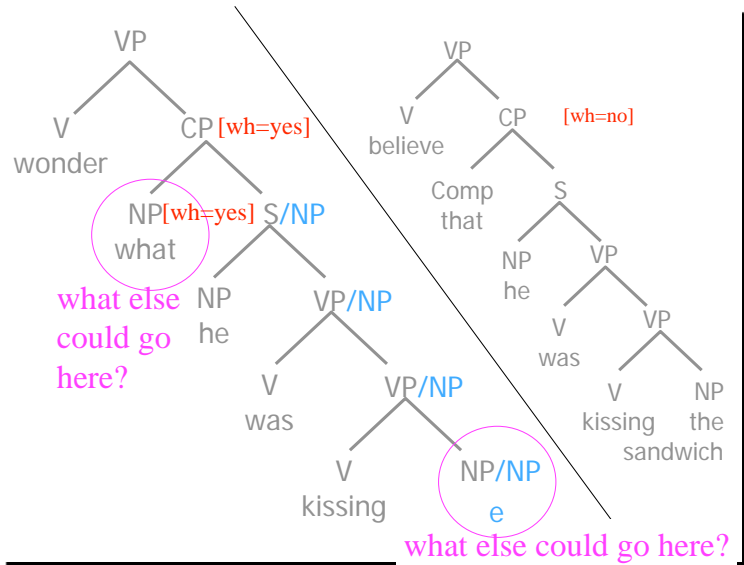
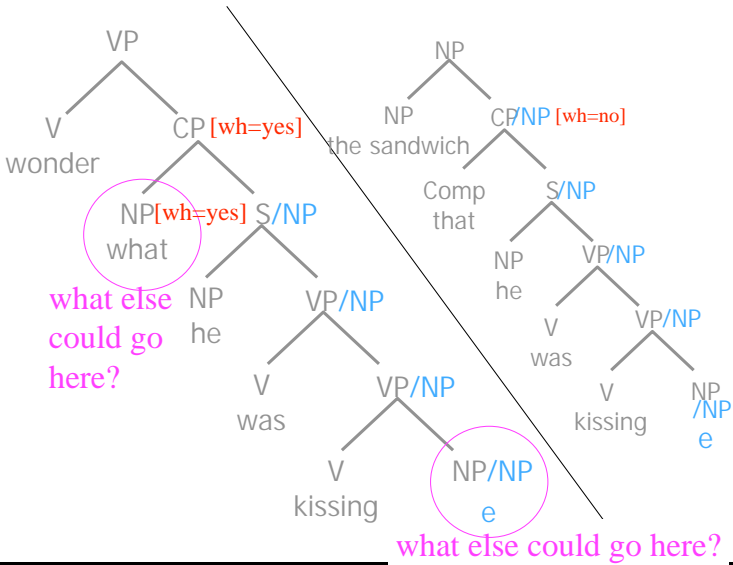
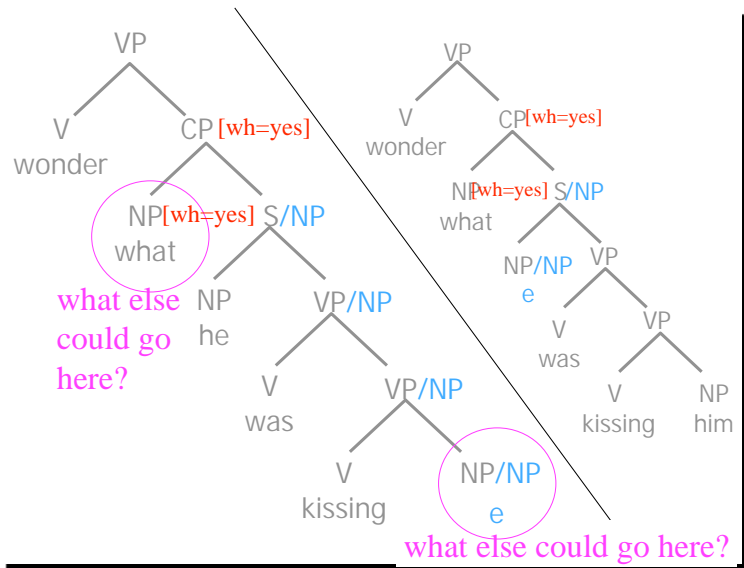
**[how could you tell the difference?]**

- **Subject gaps:**
- the sandwich that e kissed the president
- I wonder what Sally said e kissed the president
- What else has

# Gaps

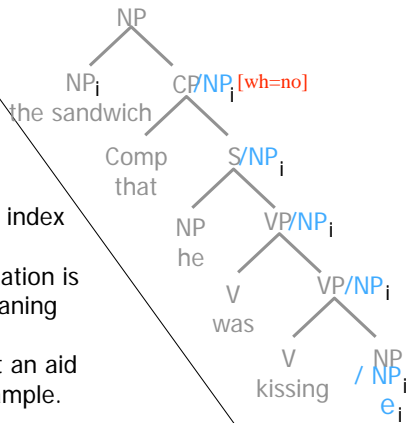
- All gaps are really the same – a missing NP:
- the sandwich that the president kissed *e*
- I wonder what Sally said the president kissed *e*
- What else has Sally consumed the pickle with *e*
- Sally consumed *e* with the pickle
- e* kissed the president
- Sally said *e* kissed the president

Phrases with missing NP:  
**X[missing=NP]**  
 or just **X/NP** for short



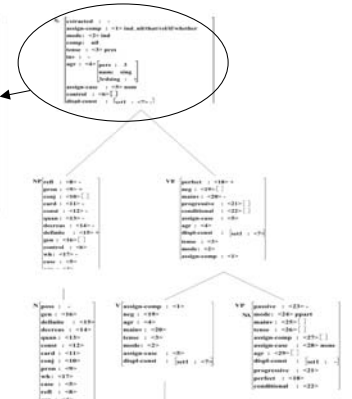
To indicate what fills a gap, people sometimes "coindex" the gap and its filler.

- Each phrase has a unique index such as "i".
- In some theories, coindexation is used to help extract a meaning from the tree.
- In other theories, it is just an aid to help you follow the example.



the money<sub>i</sub> I spend e<sub>i</sub> on the happiness<sub>j</sub> I hope to buy e<sub>j</sub>  
 which violin<sub>i</sub> is this sonata<sub>j</sub> easy to play e<sub>j</sub> on e<sub>i</sub>

```
S: [extracted : -
  assign-comp : <1> ind_nil/that/ret/id/whether
  mode: <2> ind
  comp: nil
  tense : <3> pres
  inv : -
  agr : <4> [pers : 3
            num: sing
            3rdsing : t]
  assign-case : <5> nom
  control : <6> []
  displ-const : [set1 : <7> -]
```



- Lots of features (tense, number, person, gap, vowels, commas, wh, etc...)



- Sorry, that's just how language is ...
- You know too much to write it down easily!

gone