JMORF/MORS — Morpho-Syntax

Feature Structures
Headed Rules, Trees

Francis Bond
Palacký University
https://fcbond.github.io/
bond@ieee.org

Lecture 3
Location: KB-1.11

Overview

- Review: problems with CFG
- Modeling
- Feature structures, unification (pizza)
- Features for linguistic description
- Reformulate grammar rules
- Notion of head/headedness
- Licensing of trees
- Reading the grammar fragment

Our Goals

- A descriptive, generative grammar
  - Describing English (in this case)
  - Generating all possible well-formed sentences (and no ill-formed ones)
  - Assigning appropriate structures
- Design/discover an appropriate **type** of model (through incremental improvement)
- Create a particular model (grammar fragment) for English

Problems with Context-Free Grammar

- Potentially arbitrary rules
- Gets clunky quickly with cross-cutting properties
- Not quite powerful enough for natural languages
- Solution: Replace atomic node labels (N, NP, ...) with meaningful feature structures.

Cross-cutting Grammatical Properties

	3rd singular subject	plural subject
direct object NP	*denies*	*deny*
no direct object NP	*disappears*	*disappear*

Language Models

- Two kinds of language models
  - Speakers' internalized knowledge (their grammar)
  - Set of sentences in the language
  - Probability distribution over strings
    (what "language model" usually means today: it scores sentences, but doesn't license or rule them out)

- Things involved in modelling language
  - Real world entities (utterance types)
  - Models (fully specified trees)
  - Descriptions of the models (rules, principles, lexical entries)

A delicious introduction to Typed Feature Structures

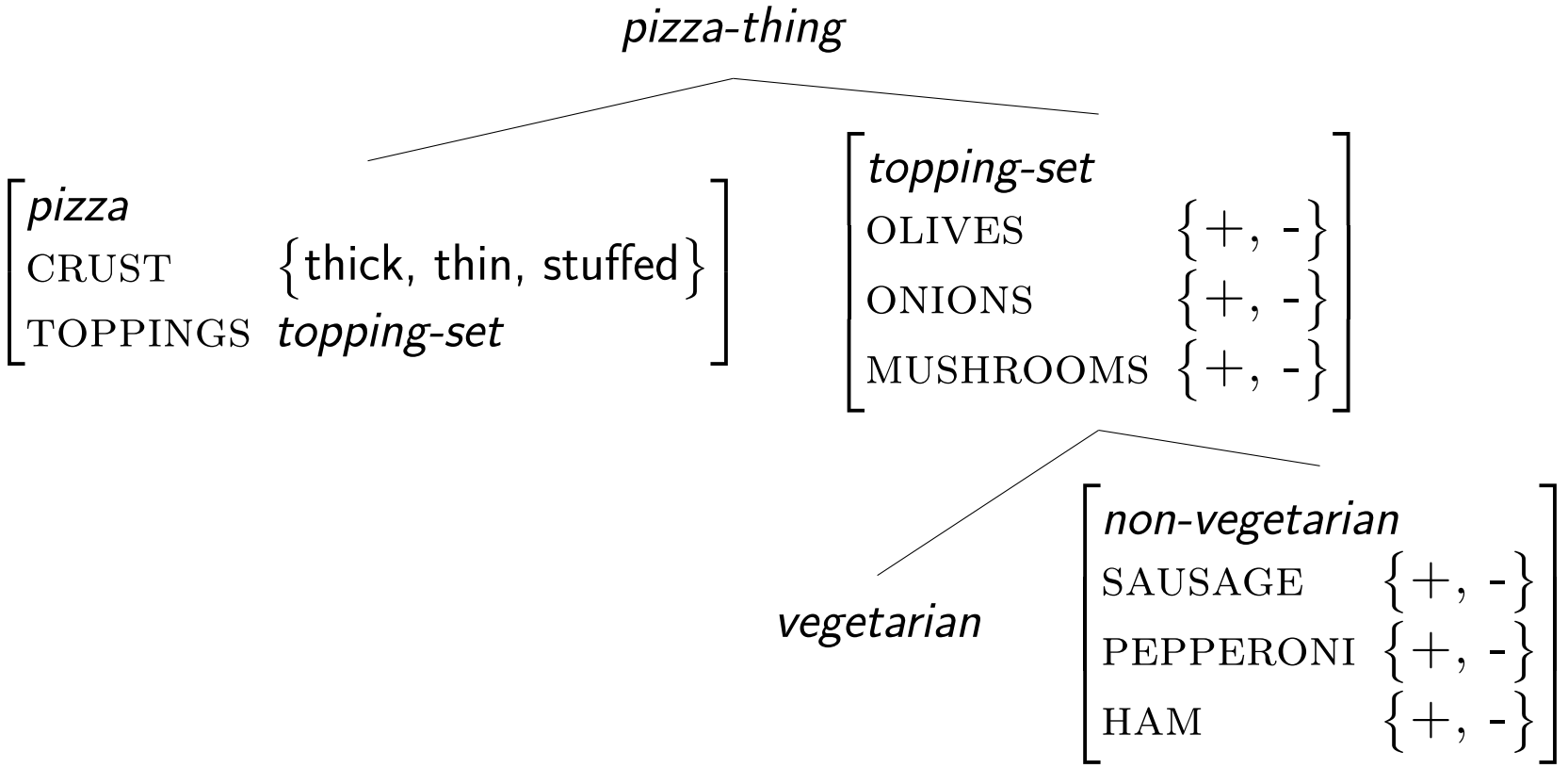
Feature Structure Descriptions

$$\begin{bmatrix} \text{FEATURE}_1 & \textit{value}_1 \\ \text{FEATURE}_2 & \textit{value}_2 \\ \quad . \ . \ . & \\ \text{FEATURE}_n & \textit{value}_n \end{bmatrix}$$

Like a form: each field (feature) has an entry (value).

Or, if you program: like a dictionary in Python (feature = key).

A Pizza Type Hierarchy



Types

Type	Features/Values	IST
pizza-thing		NONE
pizza	$\begin{bmatrix} \text{CRUST} & \{\text{thick, thin, stuffed}\} \\ \text{TOPPINGS} & \textit{topping-set} \end{bmatrix}$	*pizza-thing*
topping-set	$\begin{bmatrix} \text{OLIVES} & \{+, -\} \\ \text{ONIONS} & \{+, -\} \\ \text{MUSHROOMS} & \{+, -\} \end{bmatrix}$	*pizza-thing*
vegetarian		*topping-set*
non-vegetarian	$\begin{bmatrix} \text{SAUSAGE} & \{+, -\} \\ \text{PEPPERONI} & \{+, -\} \\ \text{HAM} & \{+, -\} \end{bmatrix}$	*topping-set*

IST: Immediate Super Type

Type Hierarchies

A type hierarchy ...

... states what kinds of objects we claim exist
— the types

... organizes the objects hierarchically into classes with shared properties
— the type hierarchy

... states what general properties each kind of object has
— the feature and feature value declarations

Pizza Descriptions and Pizza Models

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \\ \textsc{olives} & + \\ \textsc{onions} & + \end{bmatrix} \end{bmatrix}$$

How many pizza models (by definition, fully resolved) satisfy this description?

Pizza Descriptions and Pizza Models

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \\ \textsc{olives} & + \\ \textsc{onions} & + \end{bmatrix} \end{bmatrix}$$

How many pizza models (by definition, fully resolved) satisfy this description?

Two: MUSHROOMS must have a value, and it can be $+$ or $-$.

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \\ \textsc{olives} & + \\ \textsc{onions} & + \\ \textsc{mushrooms} & - \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \\ \textsc{olives} & + \\ \textsc{onions} & + \\ \textsc{mushrooms} & + \end{bmatrix} \end{bmatrix}$$

Pizzas in the World

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \\ \textsc{olives} & + \\ \textsc{onions} & + \end{bmatrix} \end{bmatrix}$$

- How many pizzas-in-the-world do the pizza models correspond to?

Pizzas in the World

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \\ \textsc{olives} & + \\ \textsc{onions} & + \end{bmatrix} \end{bmatrix}$$

- How many pizzas-in-the-world do the pizza models correspond to?
  A large, constantly-changing number.

- The 'type'/'token' distinction applies to sentences as well

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

Combining constraints is called **unification**, also written $\sqcap$. Information is added, never lost.

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \textit{pizza} \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

Combining constraints is called **unification**, also written $\sqcap$. Information is added, never lost.

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \text{CRUST} & \textit{thin} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thick} \\ \textsc{toppings} & \begin{bmatrix} \textsc{olives} & + \\ \textsc{ham} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{toppings} & \begin{bmatrix} \textsc{olives} & + \\ \textsc{onions} & + \end{bmatrix} \end{bmatrix}$$

$$= \varphi$$

A crust can't be both thick and thin, so unification fails. Failure (NULL) is also written as: $\perp, \emptyset, \phi$.

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{toppings} & \begin{bmatrix} \textsc{olives} & + \\ \textsc{ham} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \end{bmatrix} \end{bmatrix}$$

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{toppings} & \begin{bmatrix} \textsc{olives} & + \\ \textsc{ham} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{toppings} & \begin{bmatrix} \textit{vegetarian} \end{bmatrix} \end{bmatrix}$$

$$= \varphi$$

vegetarian has no feature HAM! (Even though the value is $-$.)

A New Theory of Pizzas

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{crust} & \{\text{thick, thin, stuffed}\} \\ \textsc{one-half} & \textit{topping-set} \\ \textsc{other-half} & \textit{topping-set} \end{bmatrix}$$

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{ONE-HALF} \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \text{OTHER-HALF} \begin{bmatrix} \text{OLIVES} & - \\ \text{HAM} & + \end{bmatrix} \end{bmatrix}$$

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{ONE-HALF} \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \& \begin{bmatrix} \textit{pizza} \\ \text{OTHER-HALF} \begin{bmatrix} \text{OLIVES} & - \\ \text{HAM} & + \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \textit{pizza} \\ \text{ONE-HALF} \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \\ \text{OTHER-HALF} \begin{bmatrix} \text{OLIVES} & - \\ \text{HAM} & + \end{bmatrix} \end{bmatrix}$$

Identity Constraints (tags)

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{one-half} & \begin{bmatrix} \textsc{olives} & \boxed{1} \\ \textsc{ham} & \boxed{2} \end{bmatrix} \\ \textsc{other-half} & \begin{bmatrix} \textsc{olives} & \boxed{1} \\ \textsc{ham} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

A boxed number (a **tag**) means the very same value: whatever OLIVES is on one half, it is on the other half too.

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{ONE-HALF} \quad \boxed{1} \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \\ \text{OTHER-HALF} \quad \boxed{1} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \text{OTHER-HALF} \begin{bmatrix} \text{OLIVES} & + \\ \text{MUSHROOMS} & + \end{bmatrix} \end{bmatrix}$$

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \text{ONE-HALF} \quad \boxed{1} \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \\ \text{OTHER-HALF} \quad \boxed{1} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \text{OTHER-HALF} \begin{bmatrix} \text{OLIVES} & + \\ \text{MUSHROOMS} & + \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \textit{pizza} \\ \text{ONE-HALF} \quad \boxed{1} \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \\ \text{MUSHROOMS} & + \end{bmatrix} \\ \text{OTHER-HALF} \quad \boxed{1} \end{bmatrix}$$

Mushrooms on the other half means mushrooms on both halves.

Note

$$\begin{bmatrix} \textit{pizza} & \\ \textsc{one-half} & \boxed{1} \begin{bmatrix} \textsc{olives} & + \\ \textsc{ham} & - \\ \textsc{mushrooms} & + \end{bmatrix} \\ \textsc{other-half} & \boxed{1} \end{bmatrix}$$

$$= \begin{bmatrix} \textit{pizza} & \\ \textsc{one-half} & \boxed{1} \\ & \\ \textsc{other-half} & \boxed{1} \begin{bmatrix} \textsc{olives} & + \\ \textsc{ham} & - \\ \textsc{mushrooms} & + \end{bmatrix} \\ & \end{bmatrix}$$

It doesn't matter where we write the value: both paths lead to the same thing.

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thick} \\ \textsc{one-half} & \boxed{1}\begin{bmatrix}\textsc{olives} & + \\ \textsc{onions} & -\end{bmatrix} \\ \textsc{other-half} & \boxed{1}\ \textit{vegetarian} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{one-half} & \begin{bmatrix}\textsc{olives} & + \\ \textsc{pepperoni} & +\end{bmatrix} \end{bmatrix}$$

$$= \varphi$$

Why does this fail? (Two reasons.)

Combining Constraints

$$\begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thick} \\ \textsc{one-half} & \boxed{1}\begin{bmatrix}\textsc{olives} & + \\ \textsc{onions} & -\end{bmatrix} \\ \textsc{other-half} & \boxed{1}\ \textit{vegetarian} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} \\ \textsc{crust} & \textit{thin} \\ \textsc{one-half} & \begin{bmatrix}\textsc{olives} & + \\ \textsc{pepperoni} & +\end{bmatrix} \end{bmatrix}$$

$$= \varphi$$

Why does this fail? (Two reasons.)

(1) the crust can't be both thick and thin; (2) one half is the same as the other half, which is *vegetarian*, and *vegetarian* has no PEPPERONI.

Why combine constraints?

- The pizza example illustrates how unification can be used to combine information from different sources.
- In our grammar, information will come from lexical entries, grammar rules, and general principles.
- What makes us put the information together?

Why combine constraints?

- The pizza example illustrates how unification can be used to combine information from different sources.

- In our grammar, information will come from lexical entries, grammar rules, and general principles.

- What makes us put the information together?
  The definition of a well-formed tree: every node must satisfy all of them at once.

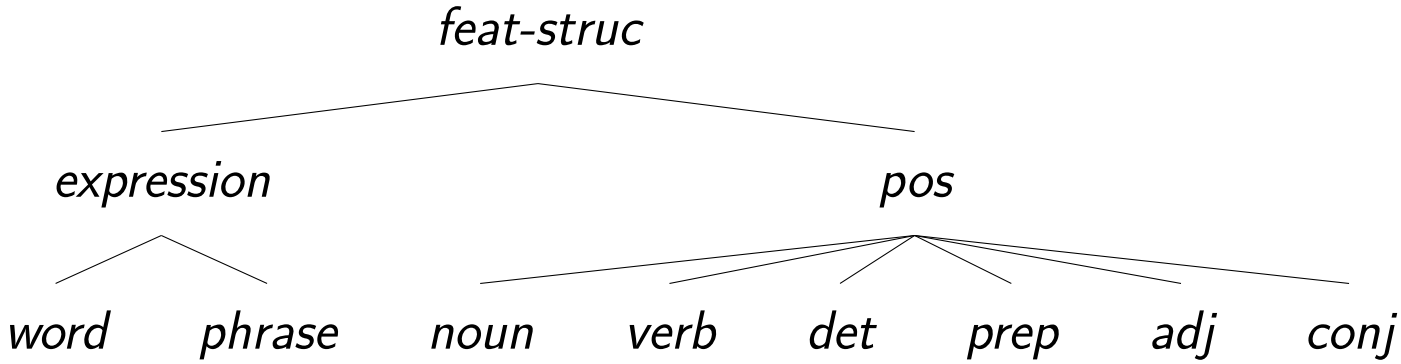
Describing Language with TFS

Linguistic Application of Feature Structures:

Making the Mnemonic Meaningful

- What do these CFG categories have in common?
  - NP & VP: are both phrases
  - N & V: are both words
  - NP & N: are both 'nouny'
  - VP & V: are both 'verby'

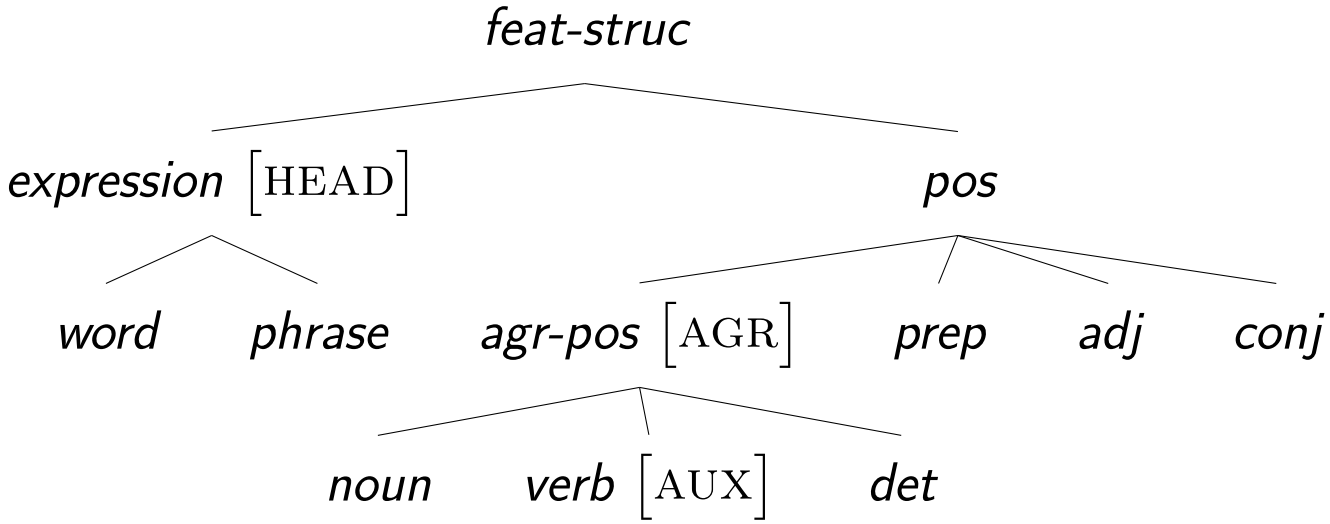
The Beginnings of Our Type Hierarchy



NP $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{noun} \end{bmatrix}$ VP $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{verb} \end{bmatrix}$

N $\left\langle \text{fly} \;,\; \begin{bmatrix} \textit{word} \\ \text{HEAD} \;\; \textit{noun} \end{bmatrix} \right\rangle$ V $\left\langle \text{fly} \;,\; \begin{bmatrix} \textit{word} \\ \text{HEAD} \;\; \textit{verb} \end{bmatrix} \right\rangle$

Type Hierarchy for Parts of Speech II



Agreement

We need more information to make words agree.

$$\left\langle \text{fly}, \begin{bmatrix} \textit{word} \\ \text{HEAD} \begin{bmatrix} \textit{noun} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} \ \textit{3rd} \\ \text{NUM} \ \textit{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{flies}, \begin{bmatrix} \textit{word} \\ \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} \ \textit{3rd} \\ \text{NUM} \ \textit{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle$$

Agreement (S → NP VP)

$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \ \boxed{1}\textit{verb} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{NP} \\ \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGR} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \ \boxed{1}\begin{bmatrix} \text{AGR} & \boxed{2} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

The values on AGR for the subject NP and verb phrase must be identical.

A Simple Feature for Valence

$$\text{iv} = \begin{bmatrix} \textit{word} \\ \text{HEAD} \;\; \textit{verb} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{itr} \end{bmatrix} \end{bmatrix} \qquad \text{tv} = \begin{bmatrix} \textit{word} \\ \text{HEAD} \;\; \textit{verb} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{str} \end{bmatrix} \end{bmatrix}$$

$$\text{dtv} = \begin{bmatrix} \textit{word} \\ \text{HEAD} \;\; \textit{verb} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{dtr} \end{bmatrix} \end{bmatrix}$$

COMPS controls how many **complements** are possible.

Head-Complement Rules (VP → V | V NP | V NP NP)

1 $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \ \boxed{1} \\ \text{VAL} \ \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} \ \boxed{1} \\ \text{VAL} \ \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$

2 $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \ \boxed{1} \\ \text{VAL} \ \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} \ \boxed{1} \\ \text{VAL} \ \begin{bmatrix} \text{COMPS} & \textit{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$ NP

3 $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \ \boxed{1} \\ \text{VAL} \ \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} \ \boxed{1} \\ \text{VAL} \ \begin{bmatrix} \text{COMPS} & \textit{dtr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$ NP NP

Underspecification

$$\text{V} = \begin{bmatrix} \textit{word} \\ \text{HEAD} \;\; \textit{verb} \end{bmatrix}$$

$$\text{VP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{verb} \end{bmatrix}$$

$$\begin{bmatrix} \text{HEAD} \;\; \textit{verb} \end{bmatrix}$$

This captures *verbiness*!

Another Valence Feature

$$\text{NP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{noun} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{itr} \\ \text{SPR} \;\; + \end{bmatrix} \end{bmatrix} \qquad \text{NOM} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{noun} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{itr} \\ \text{SPR} \;\; - \end{bmatrix} \end{bmatrix}$$

SPR controls the **specifier** (determiner and/or subject)



Specifier and Verbs

$$\text{S} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{verb} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{itr} \\ \text{SPR} \;\; + \end{bmatrix} \end{bmatrix} \qquad \text{VP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \;\; \textit{verb} \\ \text{VAL} \;\; \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \;\; \textit{itr} \\ \text{SPR} \;\; - \end{bmatrix} \end{bmatrix}$$



S is the verbal counterpart of NP, and VP the verbal counterpart of NOM.

S and NP

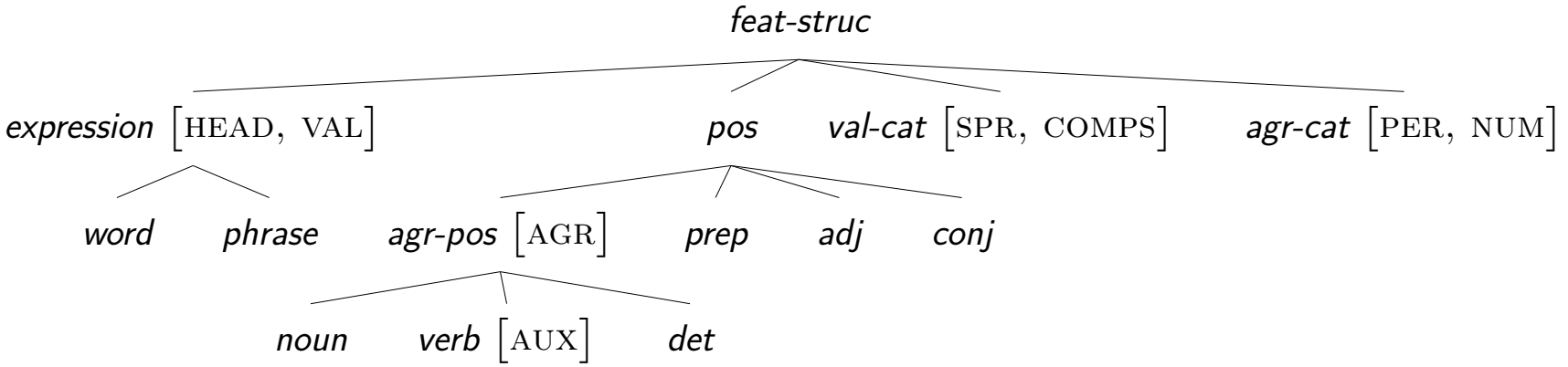
$$\begin{bmatrix} \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \textit{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

- both are fully saturated: specified and no more complements

(1) *We created a monster*

(2) *our creation of a monster*

Type Hierarchy So Far



Heads

- Intuitive idea: A phrase typically contains a word that determines its most essential properties, including
  - where it occurs in larger phrases
  - what its internal structure is
- This is called the head
- The term **head** is used both for the head word in a phrase and for all the intermediate phrases containing that word
- NB: Not all phrases have heads
  can you think of a phrase that doesn't?

Heads

- Intuitive idea: A phrase typically contains a word that determines its most essential properties, including
  - where it occurs in larger phrases
  - what its internal structure is
- This is called the head
- The term **head** is used both for the head word in a phrase and for all the intermediate phrases containing that word
- NB: Not all phrases have heads
  can you think of a phrase that doesn't?
  Coordination: ***Kim and Sandy***

Formalizing the Notion of Head

- Expressions have a feature HEAD
- HEAD's values are of type *pos* (part-of-speech)
- For HEAD values of type *agr-pos*, HEAD's value also includes the feature AGR
- Well-formed trees are subject to the **Head Feature Principle**

The Head Feature Principle

- Intuitive idea: Key properties of phrases are shared with their heads
- The **HFP**:

  In any headed phrase, the HEAD value of the mother and the head daughter must be identical.

- Sometimes described in terms of properties "percolating up" or "filtering down", but this is just metaphorical talk
- the head daughter in a headed-rule will be labeled with '**H**'.

  [*type*] $\rightarrow$ ... **H** [ ] ...

A Tree is Well-Formed if ...

- It and each subtree are licensed by a grammar rule or lexical entry
- All general principles (like the HFP) are satisfied.
- NB: Trees are part of our model of the language, so all their features have values (even though we will often be economical and leave out the values irrelevant to our current point).

Question: Unheaded Phrases

Do phrases that are not headed have HEAD features?

Question: Unheaded Phrases

Do phrases that are not headed have HEAD features?

Yes, all expressions have HEAD, but in a non-headed phrase it does not have to be identical to any daughter's.

Reformulating the Grammar Rules I

Which simple phrase structure rules (Ch 2) do these correspond to?

- Head-Complement Rule 1:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

- Head-Complement Rule 2:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \quad \text{NP}$$

Reformulating the Grammar Rules I (cont.)

- Head-Complement Rule 3:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \textbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{dtr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \quad \text{NP} \quad \text{NP}$$

Reformulating the Grammar Rules II

- Head-Specifier Rule 1:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{NP} \\ \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \textbf{H} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

- Head-Specifier Rule 2:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \text{D} \quad \textbf{H} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

Reformulating the Grammar Rules III

- Non-Branching NP Rule

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{HEAD} \ \textit{noun} \\ \text{VAL} \quad \begin{bmatrix} \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

- Head-Modifier Rule

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{phrase} \\ \text{VAL} \ \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix} \ \text{PP}$$

Reformulating the Grammar Rules III (cont.)

- Coordination Rule

$$\begin{bmatrix}\text{HEAD } \boxed{1}\end{bmatrix} \rightarrow \begin{bmatrix}\text{HEAD } \boxed{1}\end{bmatrix} + \begin{bmatrix}\textit{word} \\ \text{HEAD } \textit{conj}\end{bmatrix} \begin{bmatrix}\text{HEAD } \boxed{1}\end{bmatrix}$$

  Only coordinate two similar things!

Which Chapter 2 Rules Do the New Rules Replace?

New rule	Chapter 2 rules
Head-Complement Rule 1	
Head-Complement Rule 2	
Head-Complement Rule 3	
Head-Specifier Rule 1	
Head-Specifier Rule 2	
Non-Branching NP Rule	
Head-Modifier Rule	
Coordination Rule	

Which Chapter 2 Rules Do the New Rules Replace?

New rule	Chapter 2 rules
Head-Complement Rule 1	VP $\rightarrow$ V NOM $\rightarrow$ N
Head-Complement Rule 2	VP $\rightarrow$ V NP and PP $\rightarrow$ P NP
Head-Complement Rule 3	VP $\rightarrow$ V NP NP
Head-Specifier Rule 1	S $\rightarrow$ NP VP
Head-Specifier Rule 2	NP $\rightarrow$ D NOM (D now required)
Non-Branching NP Rule	NP $\rightarrow$ NOM, with no D (pronouns, names)
Head-Modifier Rule	VP $\rightarrow$ VP PP NOM $\rightarrow$ NOM PP
Coordination Rule	X $\rightarrow$ X^+ CONJ X

It is not one-to-one: one new rule can do the work of several old ones.

Advantages of the New Formulation

- Subject-verb agreement is stipulated only once (where?)
- Common properties of verbs with different valences are expressed by common features (for example?)
- Parallelisms across phrase types are captured (for example?)

Advantages of the New Formulation

- Subject-verb agreement is stipulated only once (where?)
  in Head-Specifier Rule 1

- Common properties of verbs with different valences are expressed by common features (for example?)
  AGR (PER, NUM): ***denies*** and ***disappears*** are both [PER 3rd, NUM sg]

- Parallelisms across phrase types are captured (for example?)
  Head-Complement Rule 2 builds both VPs and PPs

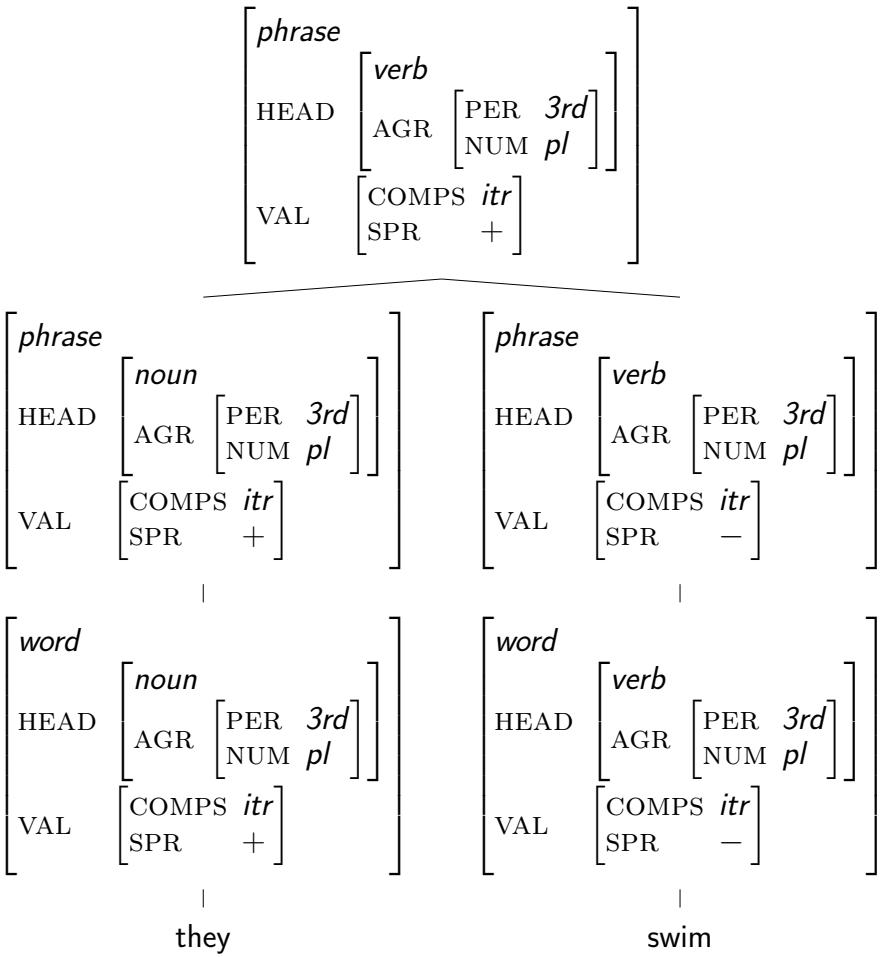
Disadvantages of the New Formulation

- We still have three head complement rules
- We still have two head specifier rules
- We only deal with three verb valences
  (Which ones? What are some others?)
- The non-branching rule doesn't really do any empirical work
- Anything else?

Disadvantages of the New Formulation

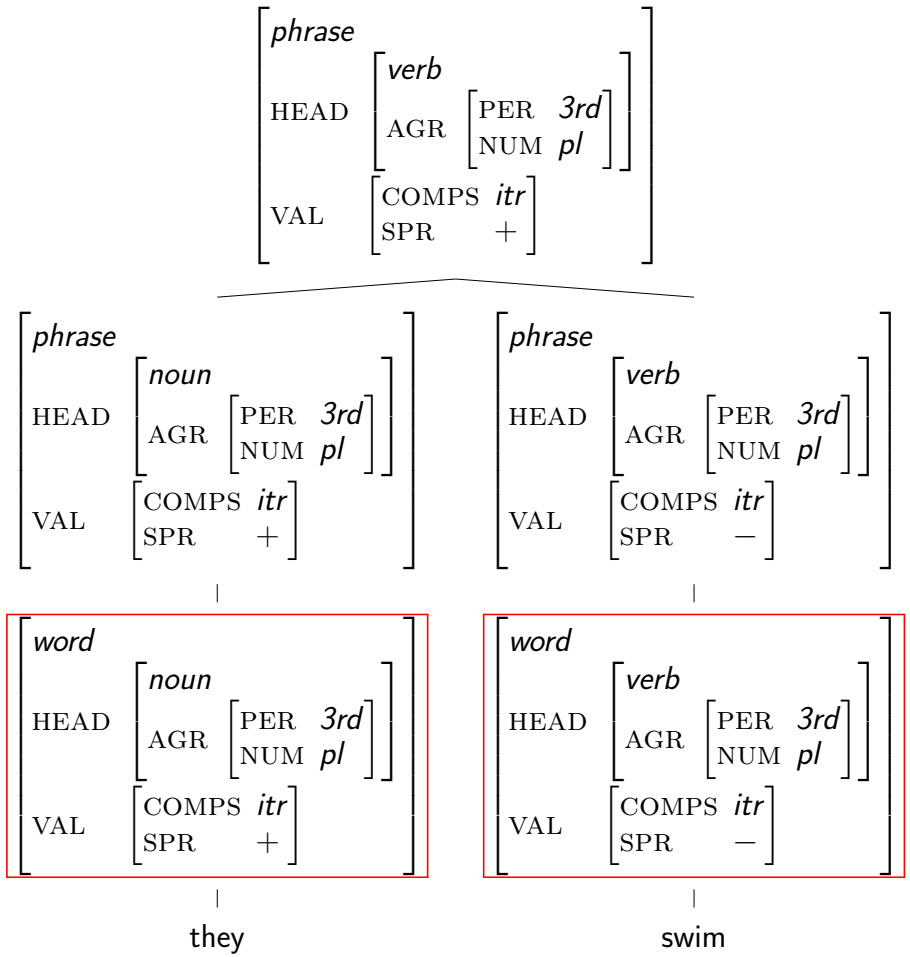
- We still have three head complement rules
- We still have two head specifier rules
- We only deal with three verb valences
  (Which ones? What are some others?)
  intransitive, transitive, ditransitive; but also ***put*** (NP PP), ***think*** (S), ***tell*** (NP S), ...
- The non-branching rule doesn't really do any empirical work
- Anything else?

Which Rule Licenses Each Node?



Which rule or lexical entry licenses each node?

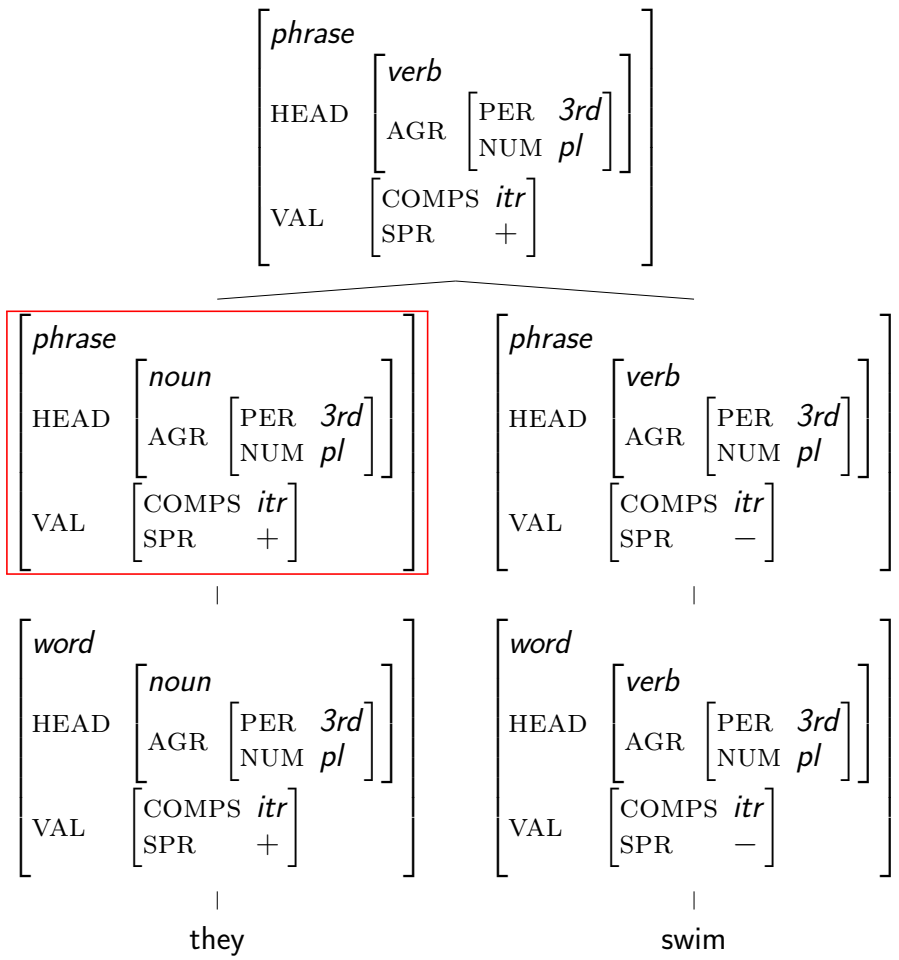
Which Rule Licenses Each Node?



Which rule or lexical entry licenses each node?

- ***they***, ***swim***: their lexical entries

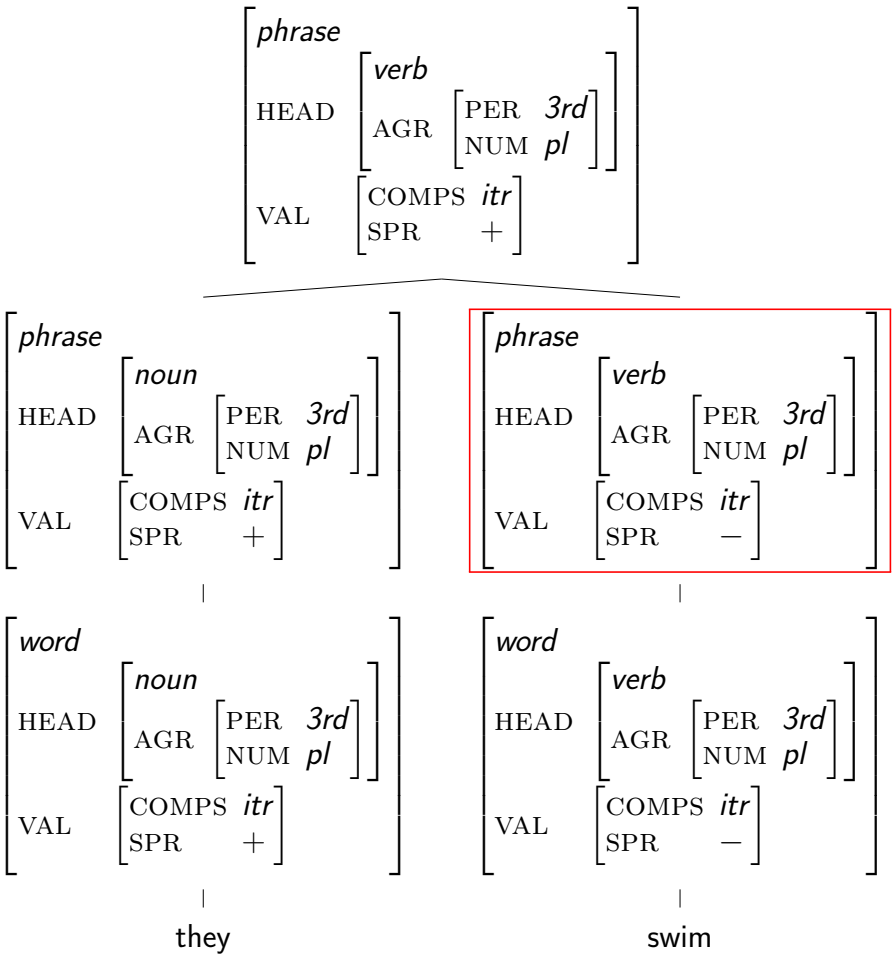
Which Rule Licenses Each Node?



Which rule or lexical entry licenses each node?

- ***they***, ***swim***: their lexical entries
- NP: the Non-Branching NP Rule (***they*** is [SPR +]); HFP

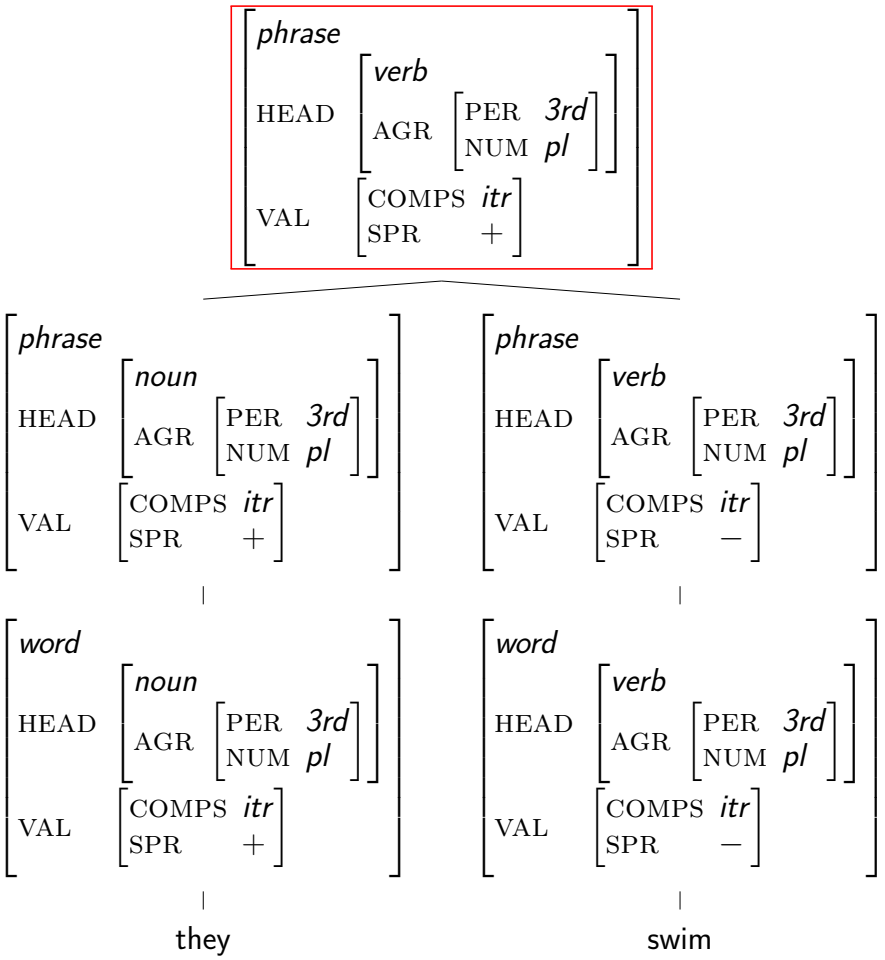
Which Rule Licenses Each Node?



Which rule or lexical entry licenses each node?

- ***they***, ***swim***: their lexical entries
- NP: the Non-Branching NP Rule (***they*** is [SPR +]); HFP
- VP: Head-Complement Rule 1 (***swim*** is [COMPS *itr*]); HFP

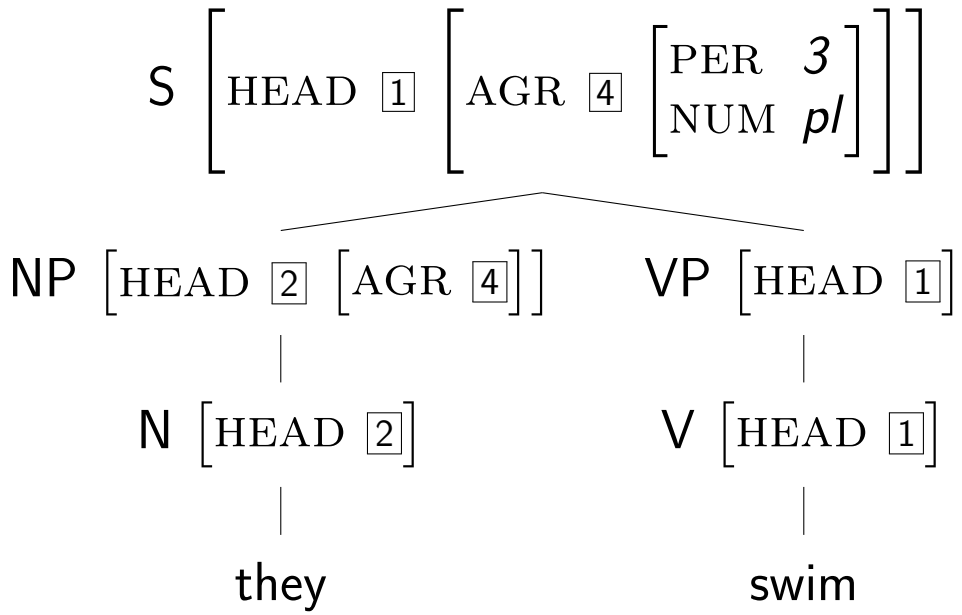
Which Rule Licenses Each Node?



Which rule or lexical entry licenses each node?

- ***they***, ***swim***: their lexical entries
- NP: the Non-Branching NP Rule (***they*** is [SPR +]); HFP
- VP: Head-Complement Rule 1 (***swim*** is [COMPS *itr*]); HFP
- S: Head-Specifier Rule 1, which also makes the NP's and VP's AGR the same; HFP

In abbreviated form



S $\Rightarrow$ *phrase*, HEAD *verb*, COMPS *itr*, SPR $+$;
VP $\Rightarrow$ *phrase*, HEAD *verb*, COMPS *itr*, SPR $-$; ...

A Question:

Since the lexical entry for swim below has only [NUM pl] as the value of AGR, how did the tree on the previous slide get [PER 3rd] in the AGR of swim?

$$\left\langle \text{swim}, \begin{bmatrix} \textit{word} & \\ \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \textit{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} & \\ \text{COMPS} & \textit{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

Answer: A Chain of Identities

This AGR value ...	is identical to ...	because of ...
they's lexical entry	***they***'s word node	the lexical entry
they's word node	the NP	the HFP
the NP	the VP	Head-Specifier Rule 1
the VP	***swim***'s word node	the HFP

So ***swim***'s AGR is the very same value as ***they***'s, which is [PER 3rd, NUM pl]. Constraints from different places combine, just like the pizza.

Reading the Grammar Fragment

Components of the Grammar

The grammar summary at the end of each chapter (and in the Appendix) has four parts:

1. The **type hierarchy**, with the features and values appropriate for each type

2. The **phrase structure rules**

3. The **well-formedness conditions** / general principles (e.g. the HFP)

4. The **lexical entries**

A tree is well-formed if it satisfies all of them.

What about the Abbreviations?

- NP, VP, S, N, V ... are just for convenience
- Each abbreviation is defined as an underspecified feature structure, e.g.

$$\text{NP} = \begin{bmatrix} \textit{phrase} & \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \textit{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

- We use them in trees as shorthand, but they do no work in the grammar

Building Trees

- You should always be able to check your trees against the grammar
- The grammar should give you enough information to fill in every feature on every node
- The book only lists a few lexical entries: find the closest one to your word and use it as a guide
- Sometimes there is a "bug" in the grammar in the book: if you can't work out a value from the grammar summary and the principles, ask!

Summary

- Review: problems with CFG; modeling
- Feature structures, unification (pizza)
- Features for linguistic description
- Reformulate grammar rules
- Notion of head/headedness
- Licensing of trees
- Reading the grammar fragment
- Next time: Valence and agreement: complex feature values

3.1 Applying the grammar

A. Formulate a lexical entry for the word *defendants*.

B. Draw a tree for the sentence *The defendants walk*. Show the values for all of the features on every node and use tags to indicate the effects of any identities that the grammar requires.

C. Explain how your lexical entry for *defendants* interacts with the Chapter 3 grammar to rule out **The defendants walks*. Your explanation should make reference to grammar rules, lexical entries and the HFP.

3.3 Determiner-Noun Agreement

The Chapter 3 grammar declares AGR to be a feature appropriate for the types *noun*, *verb*, and *det*, but so far we haven't discussed agreement involving determiners. Unlike the determiner *the*, most other English determiners do show agreement with the nouns they combine with:

(i) a bird/*a birds

(ii) this bird/*this birds

(iii) that bird/*that birds

(iv) these birds/*these bird

(v) those birds/*those bird

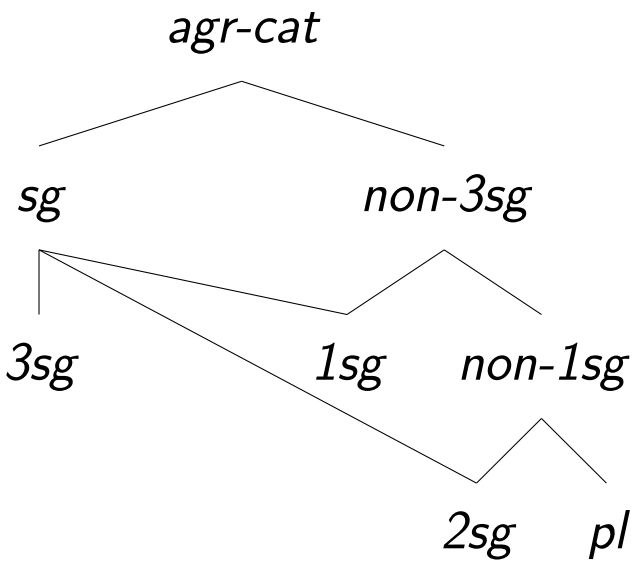
(vi) many birds/*many bird

A. Formulate lexical entries for *this* and *these*.

B. Modify Head-Specifier Rule 2 so that it enforces agreement between the noun and the determiner just like Head-Specifier Rule 1 enforces agreement between the NP and the VP.

C. Draw a tree for the NP *these birds*. Show the value for all features of every node and use tags to indicate the effects of any identities that the grammar (including your modified HSR2 and the Head Feature Principle) requires.

Types for English Agreement



Acknowledgments and References

- Course design and slides borrow heavily from Emily Bender's course: *Linguistics 566: Introduction to Syntax for Computational Linguistics* http://courses.washington.edu/ling566

- This lecture complements Sag et al. (2003, Chapter 3), and the problems are taken from it

References

Ivan A. Sag, Tom Wasow, and Emily Bender. 2003. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, second edition.