



Ling 566

Jan 4, 2019

Feature Structures
Headed Rules, Trees

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 questions

Our Goals

- 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 (atomic node labels)

- Potentially arbitrary rules
- Gets clunky quickly with cross-cutting properties
- Not quite powerful enough for natural languages

Solution: Replace atomic node labels with feature structures.

Cross-cutting Grammatical Properties

3rd singular subject

plural subject

direct object NP

denies

deny

no direct object NP

disappears

disappear

Two Kinds of Language Models

- Speakers' internalized knowledge (their grammar)
- Set of sentences in the language

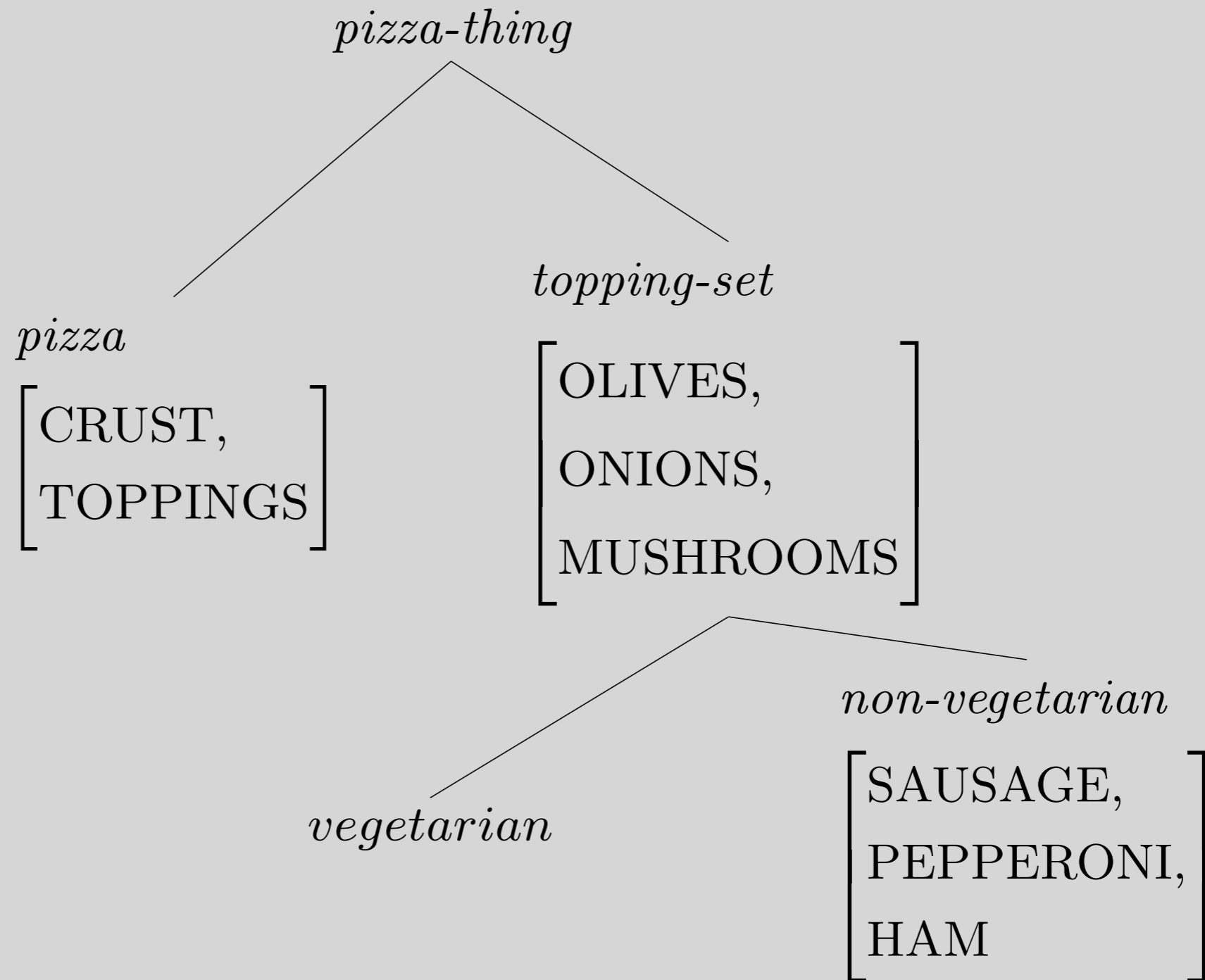
Things Involved in Modeling Language

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

Feature Structure Descriptions

FEATURE ₁	VALUE ₁
FEATURE ₂	VALUE ₂
...	
FEATURE _n	VALUE _n

A Pizza Type Hierarchy



TYPE	FEATURES/VALUES	IST
<i>pizza-thing</i>		
<i>pizza</i>	$\left[\begin{array}{ll} \text{CRUST} & \{ \text{thick, thin, stuffed} \} \\ \text{TOPPINGS} & \textit{topping-set} \end{array} \right]$	<i>pizza-thing</i>
<i>topping-set</i>	$\left[\begin{array}{ll} \text{OLIVES} & \{ +, - \} \\ \text{ONIONS} & \{ +, - \} \\ \text{MUSHROOMS} & \{ +, - \} \end{array} \right]$	<i>pizza-thing</i>
<i>vegetarian</i>		<i>topping-set</i>
<i>non-vegetarian</i>	$\left[\begin{array}{ll} \text{SAUSAGE} & \{ +, - \} \\ \text{PEPPERONI} & \{ +, - \} \\ \text{HAM} & \{ +, - \} \end{array} \right]$	<i>topping-set</i>

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

$$\left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \quad \text{thick} \\ \text{TOPPINGS} \quad \left[\begin{array}{l} \textit{vegetarian} \\ \text{OLIVES} \quad + \\ \text{ONIONS} \quad + \end{array} \right] \end{array} \right]$$

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

Answer: 2

<i>pizza</i>							
CRUST	thick						
TOPPINGS	<table><tr><td><i>vegetarian</i></td><td></td></tr><tr><td>OLIVES</td><td>+</td></tr><tr><td>ONIONS</td><td>+</td></tr></table>	<i>vegetarian</i>		OLIVES	+	ONIONS	+
<i>vegetarian</i>							
OLIVES	+						
ONIONS	+						

{<CRUST , thick> , <TOPPINGS , { <OLIVES , + > , <ONIONS, +> , <MUSHROOMS, ->}>}

{<CRUST , thick> , <TOPPINGS , { <OLIVES , + > , <ONIONS, +> , <MUSHROOMS, +>}>}

Pizza Descriptions and Pizza Models

$$\left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \quad \text{thick} \\ \text{TOPPINGS} \quad \left[\begin{array}{l} \textit{vegetarian} \\ \text{OLIVES} \quad + \\ \text{ONIONS} \quad + \end{array} \right] \end{array} \right]$$

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

Answer: A large, constantly-changing number.

Pizza Descriptions and Pizza Models

$$\left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \quad \text{thick} \\ \text{TOPPINGS} \quad \left[\begin{array}{l} \textit{vegetarian} \\ \text{OLIVES} \quad + \\ \text{ONIONS} \quad + \end{array} \right] \end{array} \right]$$

‘type’/‘token’ distinction
applies to sentences as well

Combining Constraints

$$\left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \right] \begin{array}{l} \text{thick} \\ \left[\begin{array}{l} \text{OLIVES} \\ \text{HAM} \end{array} \right] \begin{array}{l} + \\ - \end{array} \end{array} \quad \& \quad \left[\begin{array}{l} \textit{pizza} \\ \text{TOPPINGS} \end{array} \right] \left[\begin{array}{l} \text{OLIVES} \\ \text{ONIONS} \end{array} \right] \begin{array}{l} + \\ + \end{array} \right]$$

Combining Constraints

<i>pizza</i>							
CRUST	thick						
TOPPINGS	<table><tr><td>OLIVES</td><td>+</td></tr><tr><td>ONIONS</td><td>+</td></tr><tr><td>HAM</td><td>-</td></tr></table>	OLIVES	+	ONIONS	+	HAM	-
OLIVES	+						
ONIONS	+						
HAM	-						

Combining Constraints

$$\left[\begin{array}{l} \text{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \begin{array}{l} \text{thick} \\ \left[\begin{array}{l} \text{OLIVES} \\ \text{HAM} \end{array} \right] \begin{array}{l} + \\ - \end{array} \end{array} \right] \& \left[\begin{array}{l} \text{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \begin{array}{l} \text{thin} \\ \left[\begin{array}{l} \text{OLIVES} \\ \text{ONIONS} \end{array} \right] \begin{array}{l} + \\ + \end{array} \end{array} \right] \\ = \emptyset$$

Combining Constraints

$$\left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \begin{array}{l} \text{thick} \\ \left[\begin{array}{l} \text{OLIVES} \\ \text{HAM} \end{array} \right] \begin{array}{l} + \\ + \end{array} \end{array} \right] \& \left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \begin{array}{l} \text{thick} \\ \textit{vegetarian} \end{array} \right]$$

$$= \emptyset$$

Combining Constraints

$$\left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \begin{array}{l} \text{thick} \\ \left[\begin{array}{l} \text{OLIVES} \\ \text{HAM} \end{array} \right] \begin{array}{l} + \\ - \end{array} \end{array} \right] \& \left[\begin{array}{l} \textit{pizza} \\ \text{CRUST} \\ \text{TOPPINGS} \end{array} \begin{array}{l} \text{thick} \\ \textit{vegetarian} \end{array} \right]$$

$$= \emptyset$$

A New Theory of Pizzas

pizza : $\left[\begin{array}{l} \text{CRUST} \quad \left\{ \text{thick , thin , stuffed} \right\} \\ \text{ONE-HALF} \quad \textit{topping-set} \\ \text{OTHER-HALF} \quad \textit{topping-set} \end{array} \right]$

Combining Constraints

$$\begin{array}{l} \left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \end{array} \right. \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \end{array} \begin{array}{l} + \\ - \end{array} \right] \end{array} \& \begin{array}{l} \left[\begin{array}{l} \textit{pizza} \\ \text{OTHER-HALF} \end{array} \right. \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \end{array} \begin{array}{l} - \\ + \end{array} \right] \end{array} \\ \\ = \\ \begin{array}{l} \left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \\ \text{OTHER-HALF} \end{array} \right. \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \end{array} \begin{array}{l} + \\ - \end{array} \right] \\ \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \end{array} \begin{array}{l} - \\ + \end{array} \right] \end{array} \end{array}$$

Identity Constraints (tags)

<i>pizza</i>					
CRUST	thin				
ONE-HALF	<table><tr><td>OLIVES</td><td>1</td></tr><tr><td>ONIONS</td><td>2</td></tr></table>	OLIVES	1	ONIONS	2
OLIVES	1				
ONIONS	2				
OTHER-HALF	<table><tr><td>OLIVES</td><td>1</td></tr><tr><td>ONIONS</td><td>2</td></tr></table>	OLIVES	1	ONIONS	2
OLIVES	1				
ONIONS	2				

Combining Constraints

$$\left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \\ \text{OTHER-HALF} \end{array} \begin{array}{l} \boxed{1} \\ \boxed{1} \end{array} \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \end{array} \begin{array}{l} + \\ - \end{array} \right] \& \left[\begin{array}{l} \textit{pizza} \\ \text{OTHER-HALF} \end{array} \left[\begin{array}{l} \text{MUSHROOMS} \\ \text{OLIVES} \end{array} \begin{array}{l} - \\ - \end{array} \right] \right]$$

=

$$\left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \\ \text{OTHER-HALF} \end{array} \begin{array}{l} \boxed{1} \\ \boxed{1} \end{array} \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \\ \text{MUSHROOMS} \end{array} \begin{array}{l} + \\ - \\ - \end{array} \right] \right]$$

Note

$$\left[\begin{array}{l} \textit{pizza} \\ \\ \text{ONE-HALF} \\ \\ \text{OTHER-HALF} \end{array} \right] \begin{array}{l} \\ \boxed{1} \\ \\ \boxed{1} \end{array} \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \\ \text{MUSHROOMS} \end{array} \right] \begin{array}{l} + \\ - \\ - \end{array}$$

=

$$\left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \\ \\ \text{OTHER-HALF} \end{array} \right] \begin{array}{l} \boxed{1} \\ \\ \boxed{1} \end{array} \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \\ \text{MUSHROOMS} \end{array} \right] \begin{array}{l} + \\ - \\ - \end{array}$$

Combining Constraints

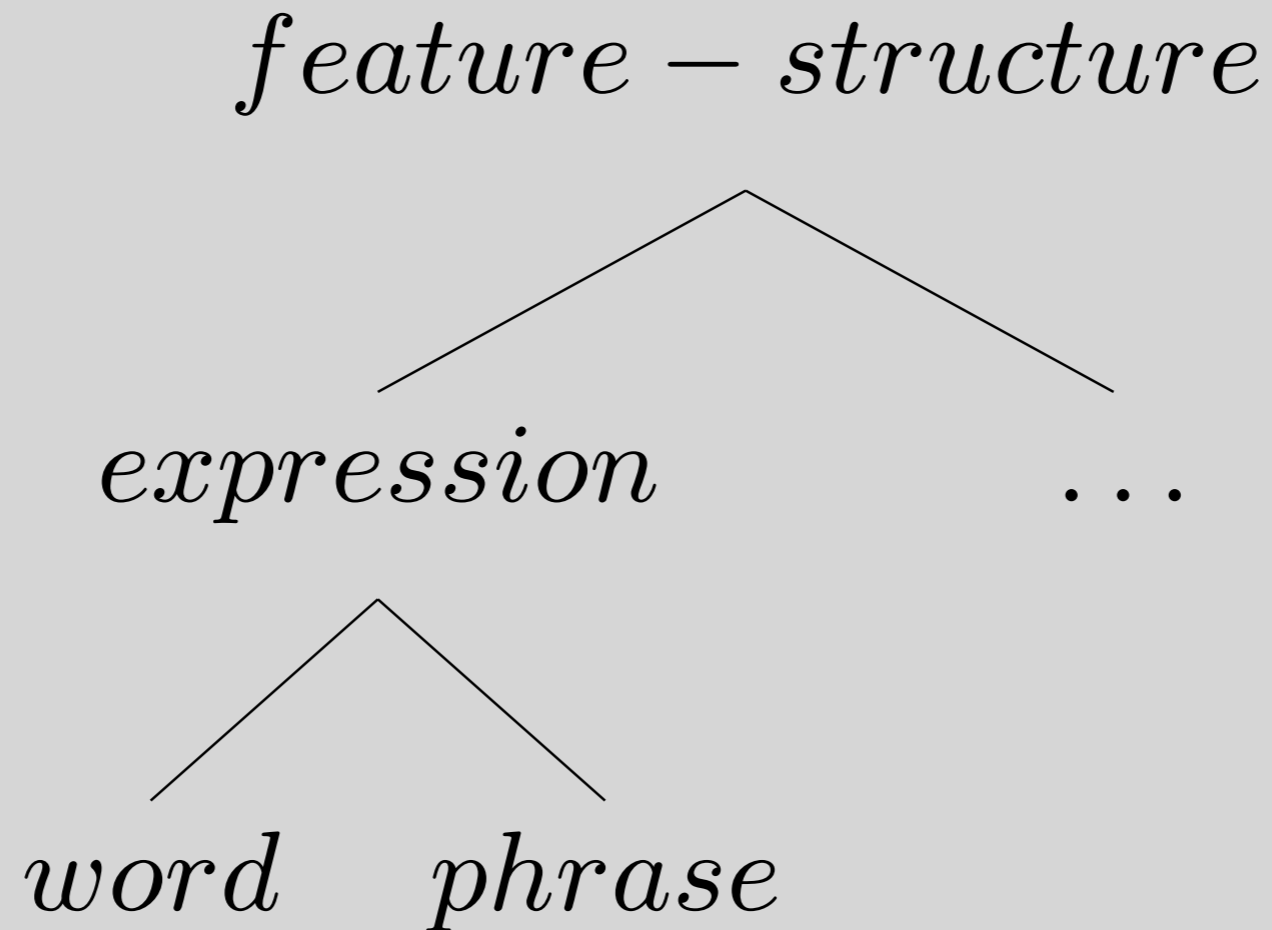
$$\left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \\ \text{OTHER-HALF} \end{array} \right] \left[\begin{array}{l} \boxed{1} \left[\begin{array}{l} \text{ONIONS} \\ \text{OLIVES} \end{array} \right] \\ \boxed{1} \textit{vegetarian} \end{array} \right] \& \left[\begin{array}{l} \textit{pizza} \\ \text{ONE-HALF} \\ \left[\begin{array}{l} \text{SAUSAGE} \\ \text{HAM} \end{array} \right] \end{array} \right]$$

$$= \emptyset$$

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.

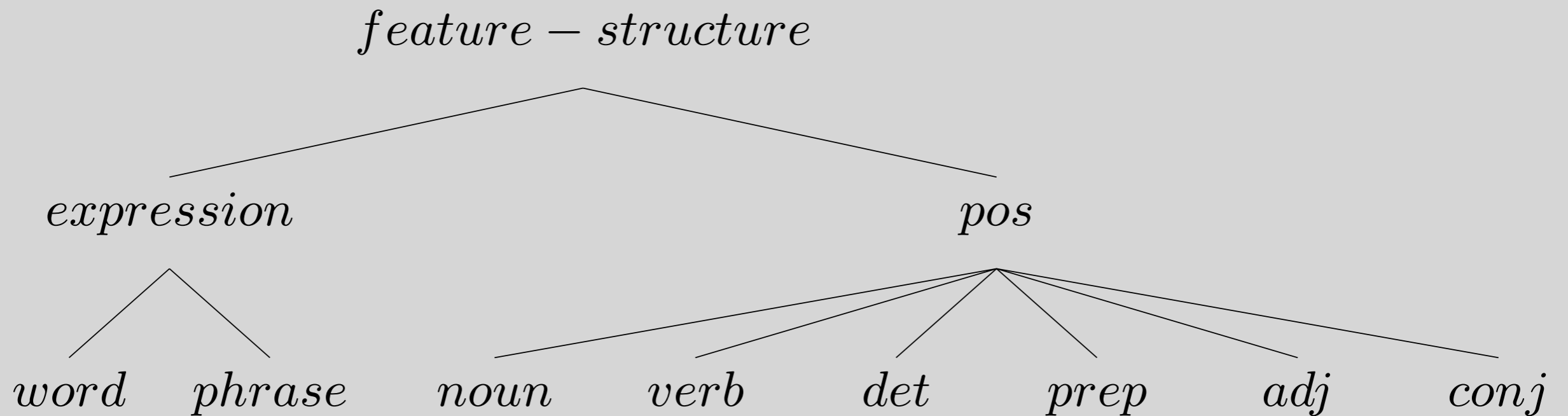
The Beginnings of Our Type Hierarchy



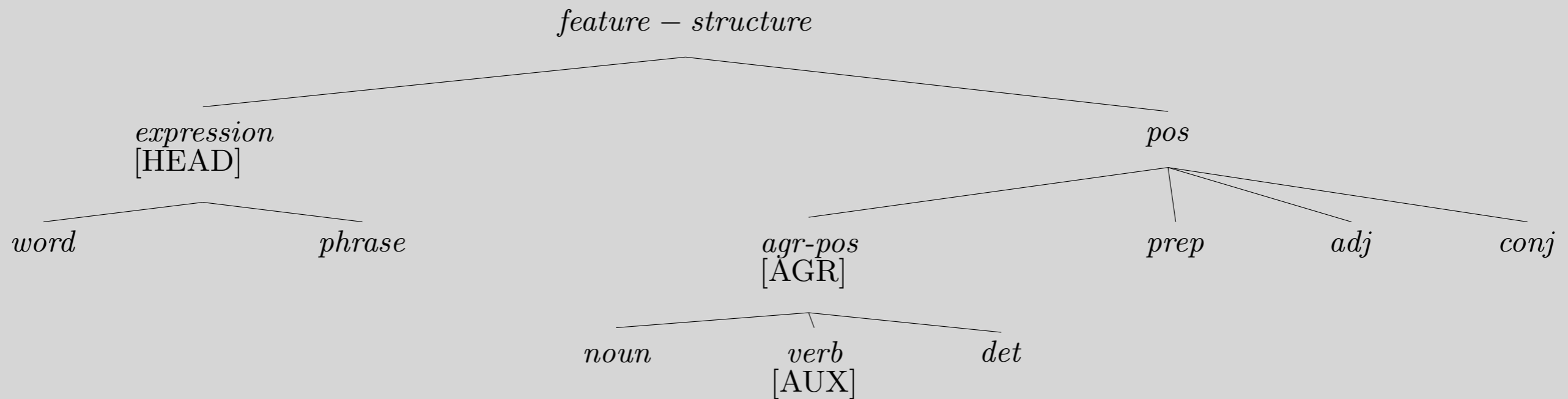
A Feature for Part of Speech

$$\text{NP} = \left[\begin{array}{l} \textit{phrase} \\ \text{HEAD} \quad \textit{noun} \end{array} \right]$$
$$\left\langle \text{bird} , \left[\begin{array}{l} \textit{word} \\ \text{HEAD} \quad \textit{noun} \end{array} \right] \right\rangle$$

Type Hierarchy for Parts of Speech I



Type Hierarchy for Parts of Speech II



A Feature for Valence

$$IV = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & [\text{COMPS} \quad \textit{itr}] \end{bmatrix}$$

$$TV = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & [\text{COMPS} \quad \textit{str}] \end{bmatrix}$$

$$DTV = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & [\text{COMPS} \quad \textit{dtr}] \end{bmatrix}$$

Underspecification

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

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

$$[\text{HEAD} \quad \textit{verb}]$$

Another Valence Feature

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

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

SPR and Verbs

$$S = \left[\begin{array}{l} \textit{phrase} \\ \text{HEAD} \quad \textit{verb} \\ \text{VAL} \quad \left[\begin{array}{l} \text{COMPS} \quad \textit{itr} \\ \text{SPR} \quad + \end{array} \right] \end{array} \right]$$

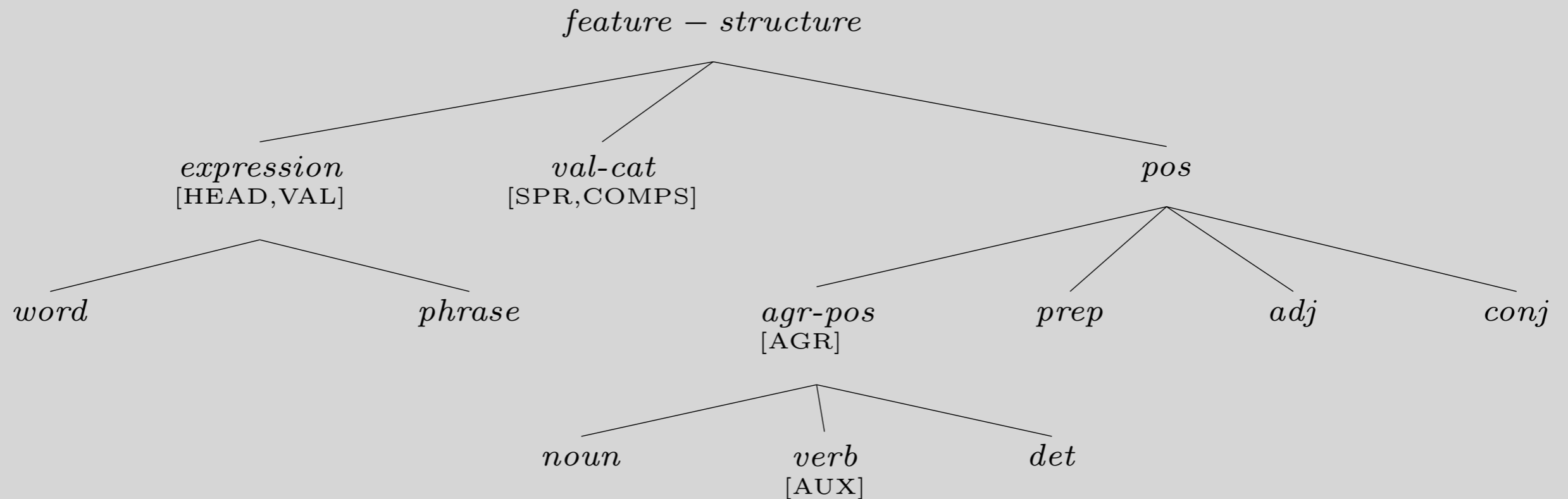
$$VP = \left[\begin{array}{l} \textit{phrase} \\ \text{HEAD} \quad \textit{verb} \\ \text{VAL} \quad \left[\begin{array}{l} \text{COMPS} \quad \textit{itr} \\ \text{SPR} \quad - \end{array} \right] \end{array} \right]$$

S and NP

$$\left[\text{VAL} \left[\begin{array}{l} \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad \quad + \end{array} \right] \right]$$

- We created a monster
- our creation of a monster

Type Hierarchy So Far



Reformulating the Grammar Rules I

Which Ch 2 rules do these correspond to?

Head-Complement Rule 1:

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS itr} \\ \text{SPR} - \end{array} \right] \end{array} \right] \rightarrow \mathbf{H} \left[\begin{array}{l} \textit{word} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS itr} \\ \text{SPR} - \end{array} \right] \end{array} \right]$$

Head Complement Rule 2:

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS itr} \\ \text{SPR} - \end{array} \right] \end{array} \right] \rightarrow \mathbf{H} \left[\begin{array}{l} \textit{word} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS str} \\ \text{SPR} - \end{array} \right] \end{array} \right] \text{ NP}$$

Head Complement Rule 3:

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS itr} \\ \text{SPR} - \end{array} \right] \end{array} \right] \rightarrow \mathbf{H} \left[\begin{array}{l} \textit{word} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS dtr} \\ \text{SPR} - \end{array} \right] \end{array} \right] \text{ NP NP}$$

Reformulating the Grammar Rules II

Head-Specifier Rule 1:

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS} \textit{ itr} \\ \text{SPR} \textit{ +} \end{array} \right] \end{array} \right] \rightarrow \left[\begin{array}{l} \text{NP} \\ \text{HEAD} \left[\begin{array}{l} \text{AGR} \boxed{1} \end{array} \right] \end{array} \right] \mathbf{H} \left[\begin{array}{l} \textit{phrase} \\ \text{HEAD} \left[\begin{array}{l} \textit{verb} \\ \text{AGR} \boxed{1} \end{array} \right] \\ \text{VAL} \left[\begin{array}{l} \text{SPR} \textit{ -} \end{array} \right] \end{array} \right]$$

Head-Specifier Rule 2:

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS} \textit{ itr} \\ \text{SPR} \textit{ +} \end{array} \right] \end{array} \right] \rightarrow \text{D} \mathbf{H} \left[\begin{array}{l} \textit{phrase} \\ \text{HEAD} \textit{ noun} \\ \text{VAL} \left[\begin{array}{l} \text{SPR} \textit{ -} \end{array} \right] \end{array} \right]$$

Reformulating the Grammar Rules III

Non-Branching NP Rule

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS} \textit{ itr} \\ \text{SPR} \textit{ +} \end{array} \right] \end{array} \right] \rightarrow \mathbf{H} \left[\begin{array}{l} \textit{word} \\ \text{HEAD} \textit{ noun} \\ \text{VAL} \left[\begin{array}{l} \text{SPR} \textit{ +} \end{array} \right] \end{array} \right]$$

Head-Modifier Rule

$$\left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{COMPS} \textit{ itr} \\ \text{SPR} \textit{ -} \end{array} \right] \end{array} \right] \rightarrow \mathbf{H} \left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \left[\begin{array}{l} \text{SPR} \textit{ -} \end{array} \right] \end{array} \right] \text{PP}$$

Coordination Rule

$$\boxed{1} \rightarrow \boxed{1}^+ \left[\begin{array}{l} \textit{word} \\ \text{HEAD} \textit{ conj} \end{array} \right] \boxed{1}$$

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?)

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
- Others?

Heads

- Intuitive idea: A phrase typically contains a word that determines its most essential properties, including
 - where it occurs in larger phrases, and
 - 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

Formalizing the Notion of Head

- Expressions have a feature HEAD
- HEAD's values are of type *pos*
- For HEAD values of type *agr-cat*, 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

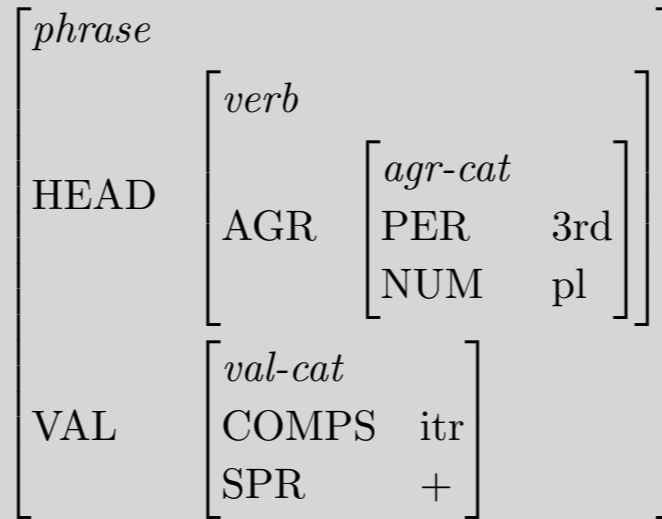
A Tree is Well-Formed if ...

- 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 lazy and leave out the values irrelevant to our current point).

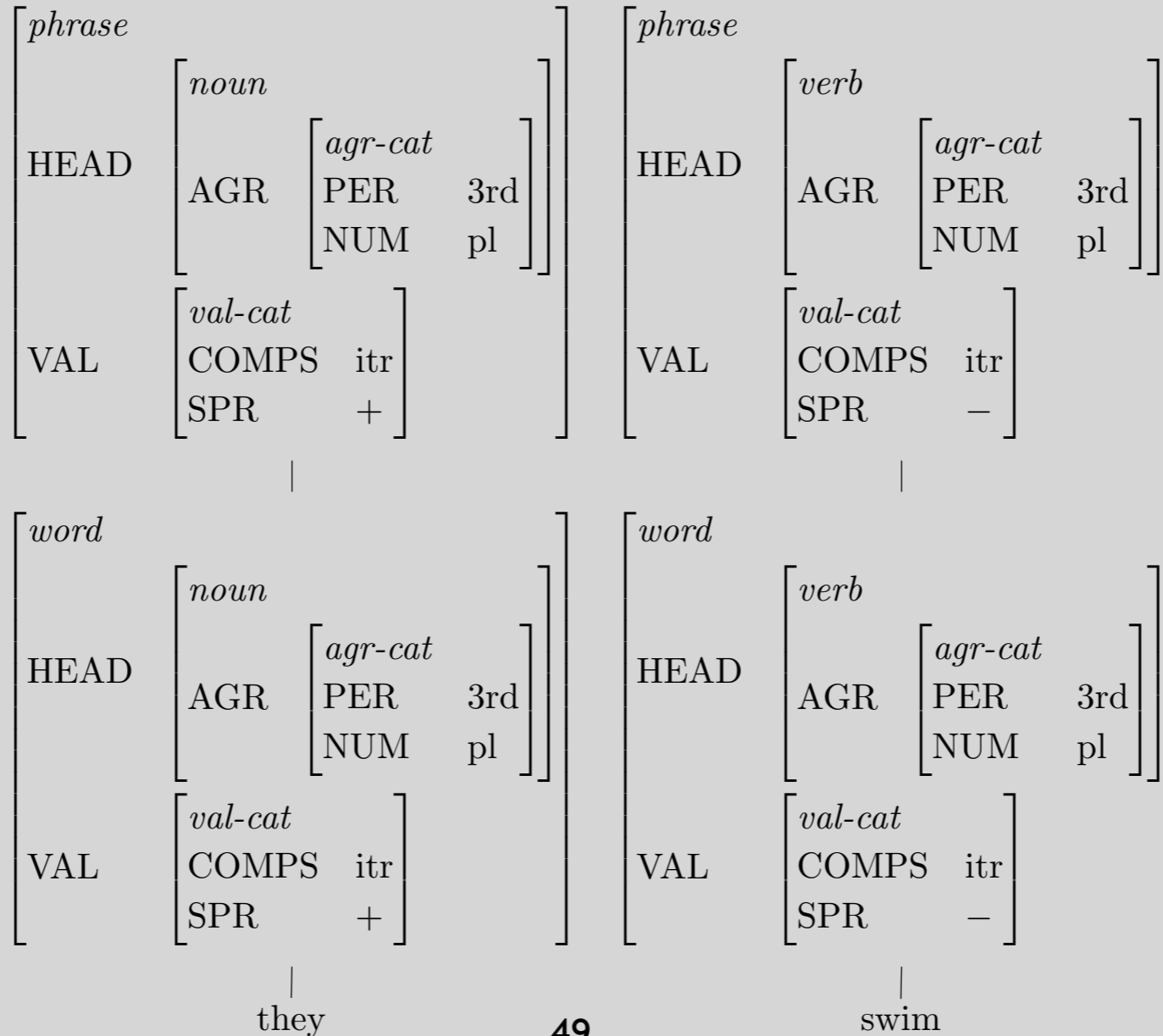
Question:

Do phrases that are not headed have
HEAD features?

Which rule licenses each node?

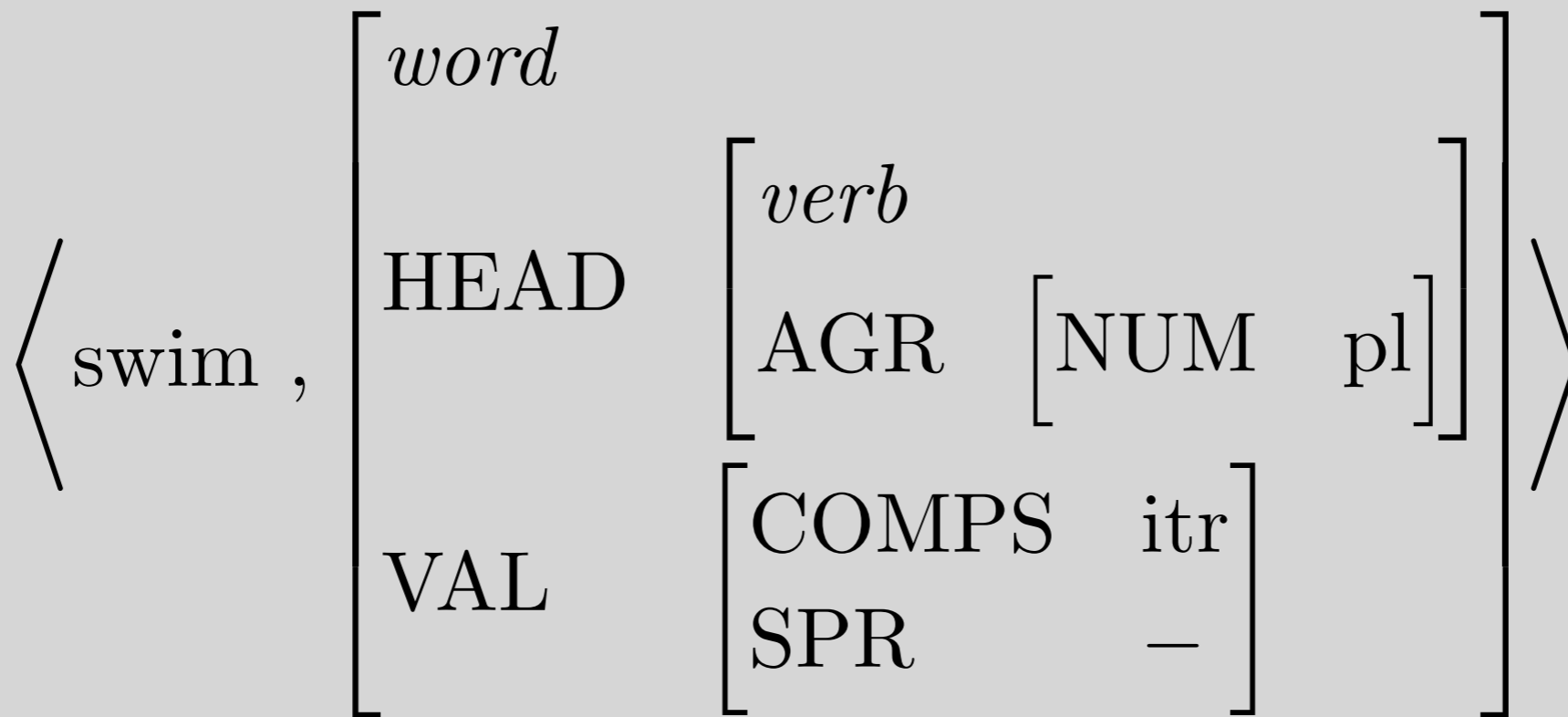


Note the three separate uses of DAGs



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?



Overview

- Review: problems with CFG
- Modeling
- Feature structures, unification (pizza)
- Features for linguistic description
- Reformulate grammar rules
- Notion of head/headedness
- Licensing of trees
- Next time: Valence and agreement

Reading Questions

- The type entity possesses the feature TEL and subsumes all other types but not all objects described in the model have a TEL feature (understandably since some of the individual type objects are no longer living). Are the features of supertypes not mandatorily inherited by their children? Or is this a simple concession of this intermediate model?
- How much information is provided in feature-structure notation seems to vary, depending on the context. Is there a general rule that dictates how much information should be provided? Is it always better to use tags to indicate identical features, or is a complete listing sometimes preferable? Further, is there a standard place to indicate what value the tag is representing?

Reading Questions

- I'm trying to understand the HFP rule, is it the method in which headed-phrases, who have identified feature values, in a sense "bring down" their values to other elements below them structurally? If a verb phrase head was identified with specific values, do the phrases/words it dominates share the same feature values because of the HFP?
- I'm trying to better understand the way HEAD works in syntax. Is there some set of rules to follow for determining the head of a phrase? Is every phrase guaranteed to have exactly one head?

Reading Questions

- When considering headed phrases, how do we identify the value of the HEAD feature for a grammar; e.g., given $S \rightarrow NP VP$, how do we identify whether NP or VP is the head daughter? Does our grammar define this from the top-down or from the bottom-up? The book states "[...] an NP structure is nominal because it has an N inside of it. That N is the head daughter of the NP structure." -- this makes sense, but I'm not sure how to generalize this.

Reading Questions

- Does calling NOM and VP incomplete (as opposed to "complete" S and NP) have a linguistic basis or is that just how we look at them in this grammar theory?
- Why do determiners have 'itr' as the value for the COMPS feature? Doesn't it always need a NOM to follow it, at least an implied one? Do we just not consider the following NOM a complement?

Reading Questions

- I'm having a bit of trouble understanding the COMPS feature. For example, I'm not sure how it works with a verb like “danced” that has the same form yet can take more than one complement or none at all (as in the example sentences below). How is the COMPS feature written for such words?
 - John danced.
 - John danced the tango.
 - John danced the tango with Mary.
- As the text says, we can call subject NPs “specifiers” (p. 64), but nothing in the feature structures for them actually shows this, right? It's just the negative SPR value for VPs that implies that NPs are specifiers?

Reading Questions

- I don't see any other rules allowing for the addition of "the film" to "Alex saw the movie on Tuesday" so does that mean a lack of specification for COMPS mean I can simply glom whatever I want on to the phrase?
- Why does leaving [COMPS itr] off of phrases cause problems? (See Ex 3)

Reading Questions

- This seems like a lot of work! Do features for languages always have to be developed manually, or is there a way to automate some/all aspects of the process?
- It seems like this approach does make it easier to generate questions, but it seems like this type of approach could easily lead to an almost infinitely complex feature list rather than an infinitely long list of acceptable types of words. How do we manage to keep the feature numbers under control? Or is this a drawback of the approach?
- How do we determine that a feature list is "good enough" for our grammar? How would we make sure to avoid creating a feature list that overfits for the sentences we create our grammar off of?