CS447: Natural Language Processing
http://courses.engr.illinois.edu/cs447

Lecture 19: Feature structures and unification

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center

Today's lecture

Feature structures form the basis for many grammar formalisms used in computational linguistics.

Feature structure grammars (aka attribute-value grammars, or unification grammars) can be used as

- a more compact way of representing rich CFGs
- a way to represent more expressive grammars

Simple grammars overgenerate

$$
\begin{array}{rcl}
S & \rightarrow & NP \quad VP \\
VP & \rightarrow & Verb \quad NP \\
NP & \rightarrow & Det \quad Noun \\
Det & \rightarrow & the \mid a \mid these \\
Verb & \rightarrow & eat \mid eats \\
Noun & \rightarrow & cake \mid cakes \mid student \mid students
\end{array}
$$

This generates ungrammatical sentences like
"these student eats a cakes"

We need to capture (number/person) agreement

Refining the nonterminals

$$
\begin{array}{rcl}
S & \longrightarrow & NPsg \;\; VPsg \\
S & \longrightarrow & NPpl \;\; VPpl \\
VPsg & \longrightarrow & VerbSg \;\; NP \\
VPpl & \longrightarrow & VerbPl \;\; NP \\
NPsg & \longrightarrow & DetSg \;\; NounSg \\
DetSg & \longrightarrow & the \mid a \\
\ldots & \ldots & \ldots
\end{array}
$$

This yields very large grammars.

- What about person, case, ...?

Difficult to capture generalizations.

- Subject and verb have to have number agreement
- $NPsg$, $NPpl$ and NP are three distinct nonterminals

Feature structures

Replace atomic categories with feature structures:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix} \qquad \begin{bmatrix} \text{CAT} & \text{VP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{VFORM} & \text{FINITE} \end{bmatrix}$$

A feature structure is a list of features (= attributes), e.g. CASE, and values (eg NOM).

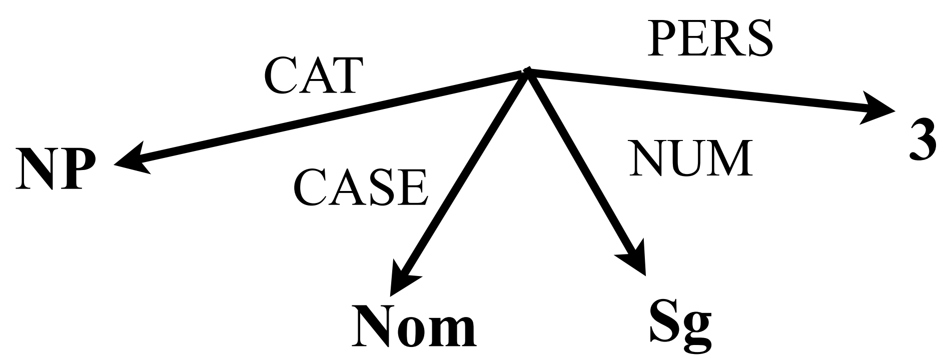
We often represent feature structures as attribute value matrices (AVM)

Usually, values are typed (to avoid CASE:SG)

Feature Structures: The Basics

Feature structures as directed graphs

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix} =$$



Complex feature structures

We distinguish between atomic and complex feature values.
A complex value is a feature structure itself.

This allows us to capture better generalizations.

Only atomic values:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix}$$

Complex values:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix} \end{bmatrix}$$

Feature paths

$$\textbf{NP:}\ \left[\begin{array}{ll} \text{CAT} & \text{NP} \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{array}\right] \end{array}\right]$$

A feature path allows us to identify particular values in a feature structure:

$\langle \textbf{NP}\ \text{CAT} \rangle = \text{NP}$

$\langle \textbf{NP}\ \text{AGR CASE} \rangle = \text{NOM}$

Unification

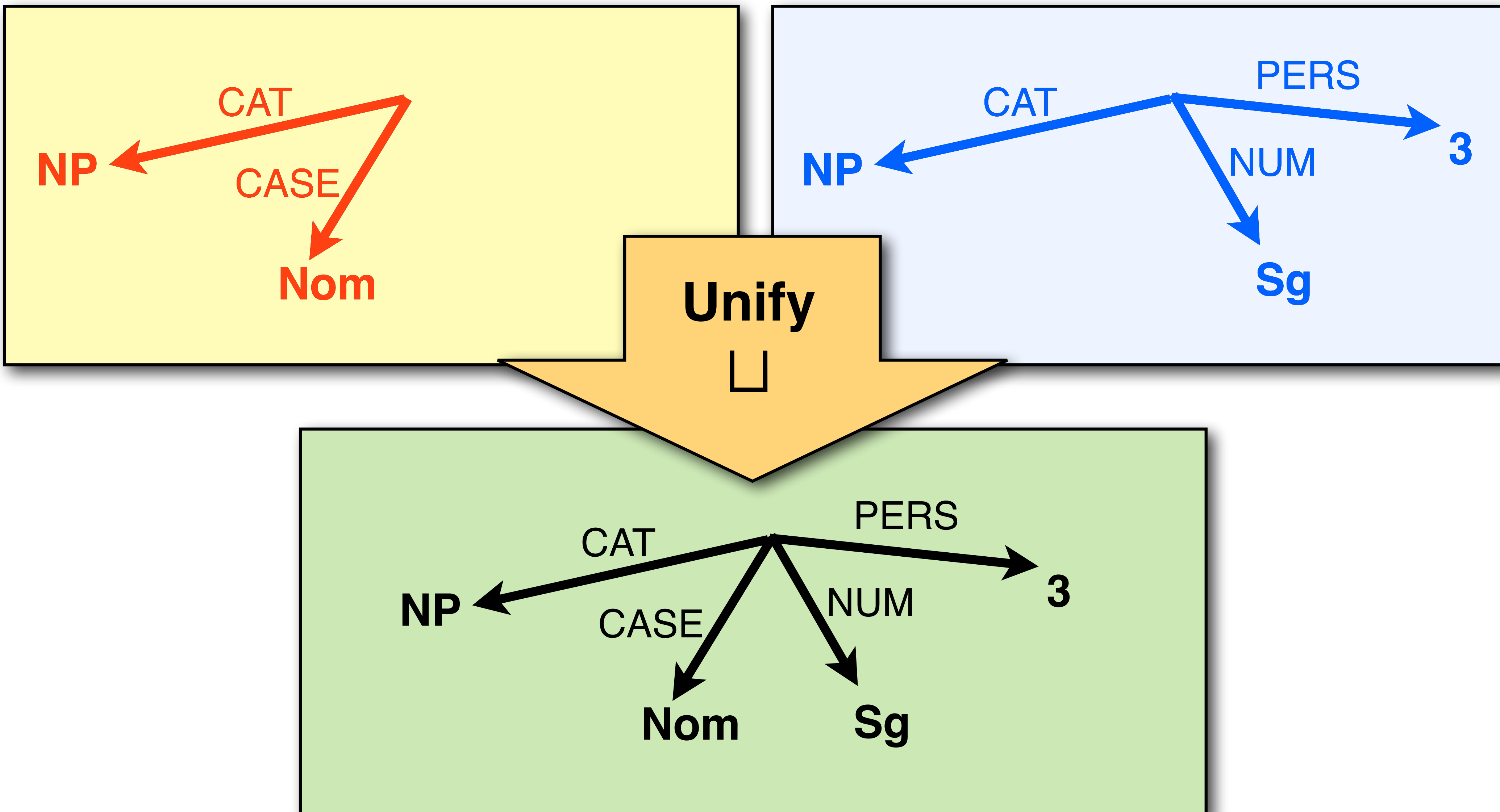
Two feature structures A and B **unify** (A ⊔ B)
if they can be merged into one consistent feature structure C:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{CASE} & \text{NOM} \end{bmatrix} \sqcup \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{PERS} & 3 \end{bmatrix} = \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix}$$

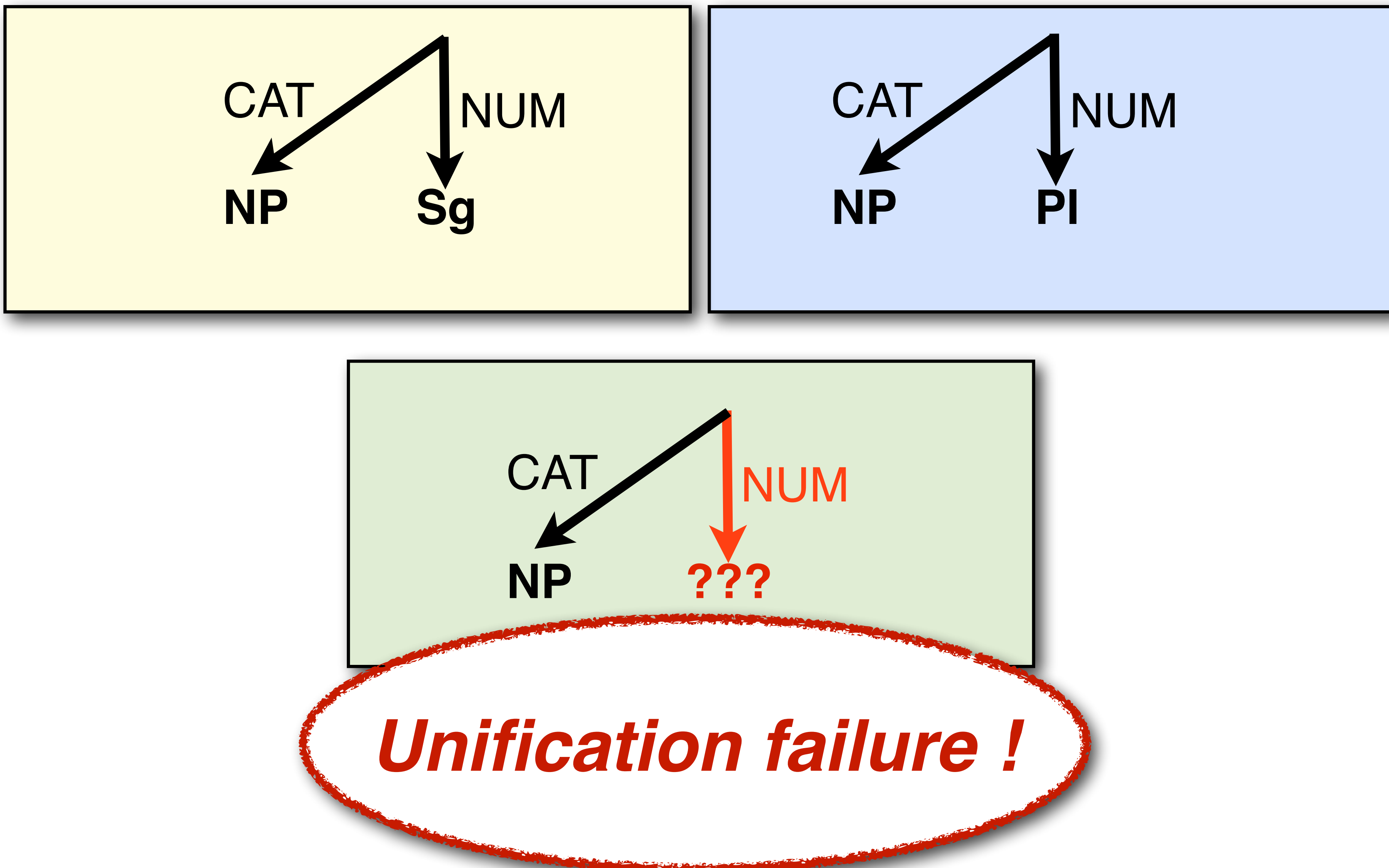
Otherwise, unification fails:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{CASE} & \text{NOM} \end{bmatrix} \sqcup \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{PL} \end{bmatrix} = \emptyset$$

Unification as graph-matching



Unification as graph-matching



Feature Structure Grammars

PATR-II style feature structures

CFG rules are augmented with constraints:

$$\mathbf{A_0 \rightarrow A_1 \ldots A_n}$$
$$\{\text{set of constraints}\}$$

There are two kinds of constraints:

Unification constraints:

$$\langle \mathbf{A_i} \text{ feature-path} \rangle = \langle \mathbf{A_j} \text{ feature-path} \rangle$$

Value constraints:

$$\langle \mathbf{A_i} \text{ feature-path} \rangle = \text{ atomic value}$$

A grammar with feature structures

S → **NP VP**				**Grammar rule**
	$\langle$**NP** *NUM*$\rangle$	=	$\langle$**VP** *NUM*$\rangle$	**Constraints**
	$\langle$**NP** *CASE*$\rangle$	=	*nom*	

NP → **DT NOUN**				**Grammar rule**
	$\langle$**NP** *NUM*$\rangle$	=	$\langle$**NOUN** *NUM*$\rangle$	**Constraints**
	$\langle$**NP** *CASE*$\rangle$	=	$\langle$**NOUN** *CASE*$\rangle$	

NOUN → *cake*				**Lexical entry**
	$\langle$**NOUN** *NUM*$\rangle$	=	*sg*	**Constraints**

With complex feature structures

S	$\rightarrow$	**NP VP**			**Grammar rule**
		⟨**NP** *AGR*⟩	=	⟨**VP** *AGR*⟩	**Constraints**
		⟨**NP** *CASE*⟩	=	*nom*	

NP	$\rightarrow$	**DT NOUN**			**Grammar rule**
		⟨**NP** *AGR*⟩	=	⟨**NOUN** *AGR*⟩	**Constraints**

NOUN	$\rightarrow$	*cake*			**Lexical entry**
		⟨**NOUN** AGR NUM⟩	=	*sg*	**Constraints**

Complex feature structures capture better generalizations (and hence require fewer constraints) — cf. the previous slide

The head feature

Instead of implicitly specifying heads for each rewrite rule, let us define a **head feature**.

The head of a VP has the same agreement feature as the VP itself:

$$\begin{bmatrix} \text{CAT} & \text{VP} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{HEAD} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

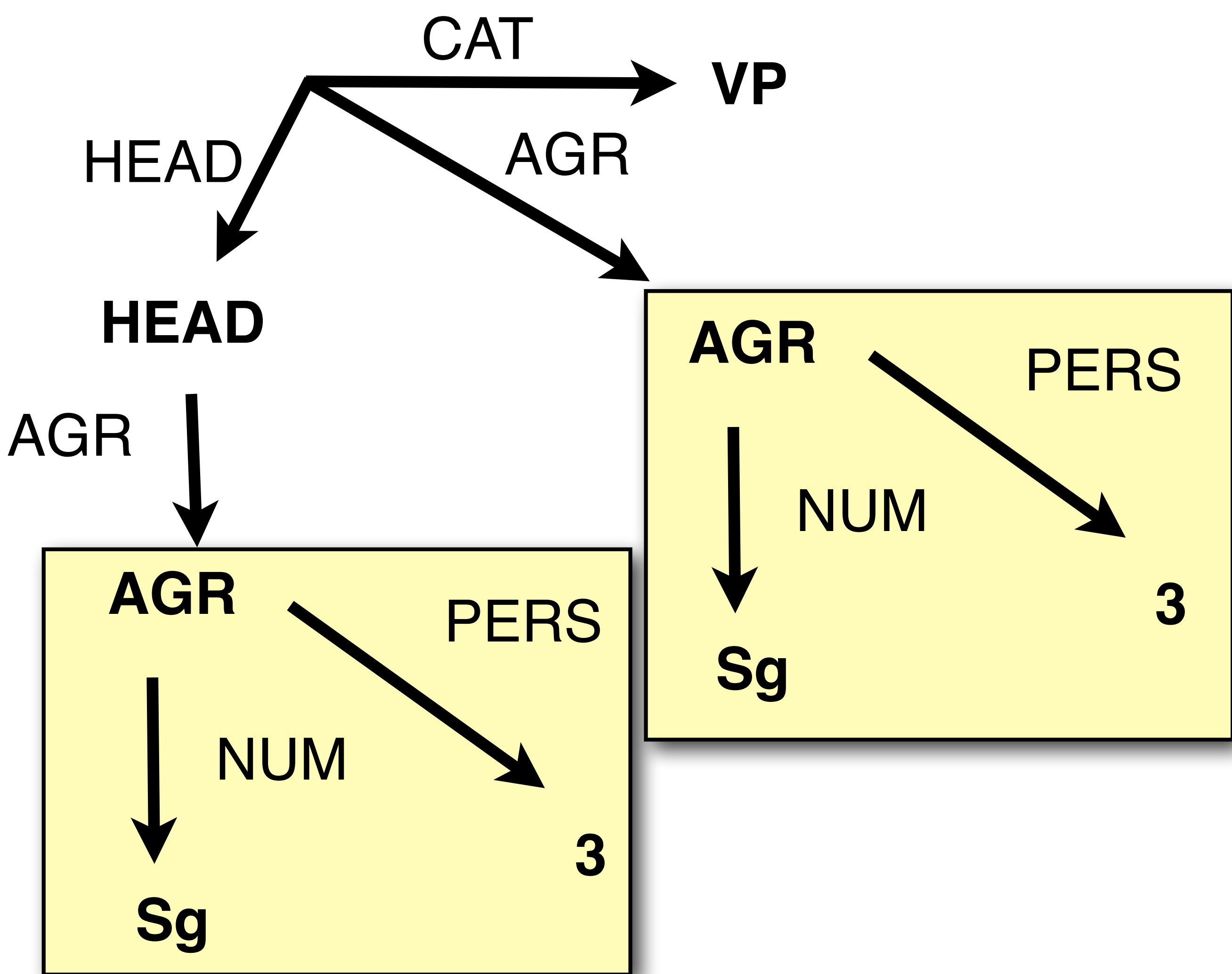
Re-entrancies

What we *really* want to say is that the agreement feature of the head is *identical* to that of the VP itself.

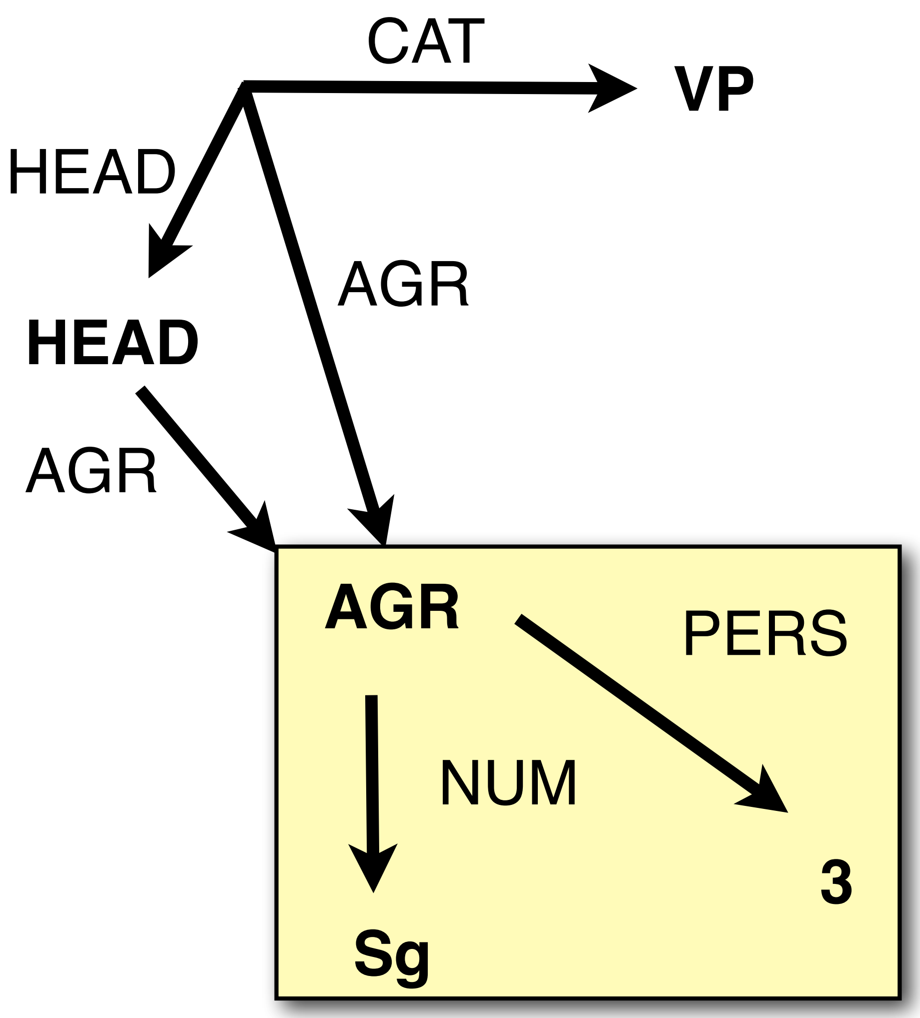
This corresponds to a re-entrancy in the FS (indicated via coindexation $\boxed{1}$)

$$
\begin{bmatrix}
\text{CAT} & \text{VP} \\
\text{AGR} & \boxed{1} \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \\
\text{HEAD} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix}
\end{bmatrix}
$$

Re-entrancies - not like this:



Re-entrancies - but like this:



Extensions of feature structures

Disjunction:

eats: [PERS 3], *eat:* [PERS: $1 \vee 2$]

Negation:

eats: [PERS 3] *eat:* [PERS: $\neg 3$]

List-valued features:

English *give* takes an NP and a to-PP as arguments, and they have to appear in a specific order:

"give the book to you" not: **"give to you the book"*

give: [CAT: VP, SUBCAT: <NP, PPto>]

Set-valued features:

German *geben* takes three NPs, which can appear in any order:

ich gebe dir das Buch | das Buch gebe ich dir | dir gebe ich das Buch,...

geben: [CAT: S, SUBCAT {NPnom NPacc, NPdat}]

The Expressive Power of Feature Structure Grammars

Attribute-Value Grammars and CFGs

If every feature can only have a finite set of values, any attribute-value grammar can be compiled out into a (possibly huge) context-free grammar

Going beyond CFGs

The power-of-2 language: $L_2 = \{w^i \mid i \text{ is a power of } 2\}$

L_2 is a (fully) context-sensitive language.

(*Mildly* context-sensitive languages have the **constant growth property** (the length of words always increases by a constant factor c))

Here is a feature grammar which generates L_2:

$$
\begin{aligned}
\mathrm{A} \rightarrow \quad & a \\
& \langle \mathrm{A}\ F \rangle = 1 \\
\mathrm{A} \rightarrow \quad & \mathrm{A}_1 \quad \mathrm{A}_2 \\
& \langle \mathrm{A}\ F \rangle = \langle \mathrm{A}_1 \rangle \\
& \langle \mathrm{A}\ F \rangle = \langle \mathrm{A}_2 \rangle
\end{aligned}
$$

What do feature structures represent?

Using feature structures (I)

We have just seen how to use feature structures to refine/extend context-free grammars.

CFGs provide a *procedural* way to define a language:

- The grammar provides a set of rewrite rules.

- The language consists of the set of terminal strings (the subset of Σ^*, the set of all strings over the vocabulary Σ) that can be obtained via a sequence of rewrite rules from the start symbol S:

  Rewrite S as NP VP, rewrite NP as DT Noun, rewrite VP as…

Using feature structures (II)

We can also view feature structures as a *declarative* way to specify a language:

- Assume the 'universe' of linguistic objects is Σ^* (the set of all strings over the vocabulary Σ)
- The grammar specifies a set of feature structures.
- Each feature structure specifies a set of constraints over linguistic objects.
  Hence, each feature structures defines a set of terminal strings (a subset of Σ^*) that obeys these constraints.
- The language consists of the set of terminal strings that are allowed by at least one feature structure.

Features as constraints

Features impose **constraints** on linguistic objects.

If A and B unify, but B contains more features than A, B is more specific than A:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{CASE} & \text{NOM} \end{bmatrix} \qquad \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix}$$

A **B**

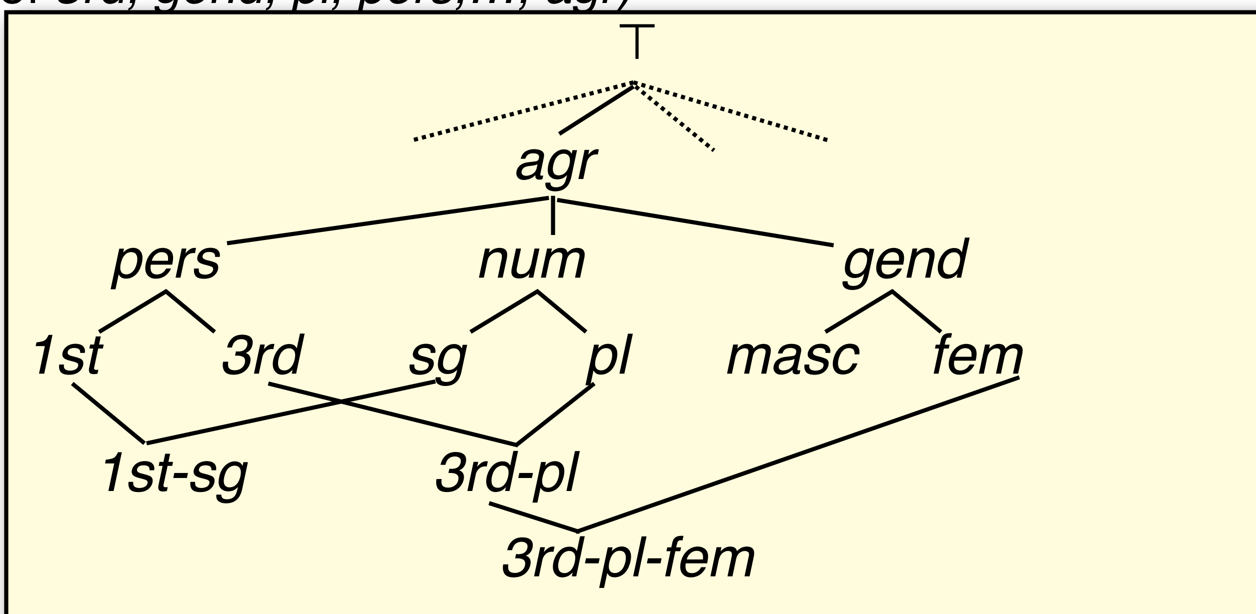
We also say that A **subsumes** the more specific B.
Subsumption defines a *partial ordering*
over feature structures.

Typed feature structures

In a **typed feature structure** system,

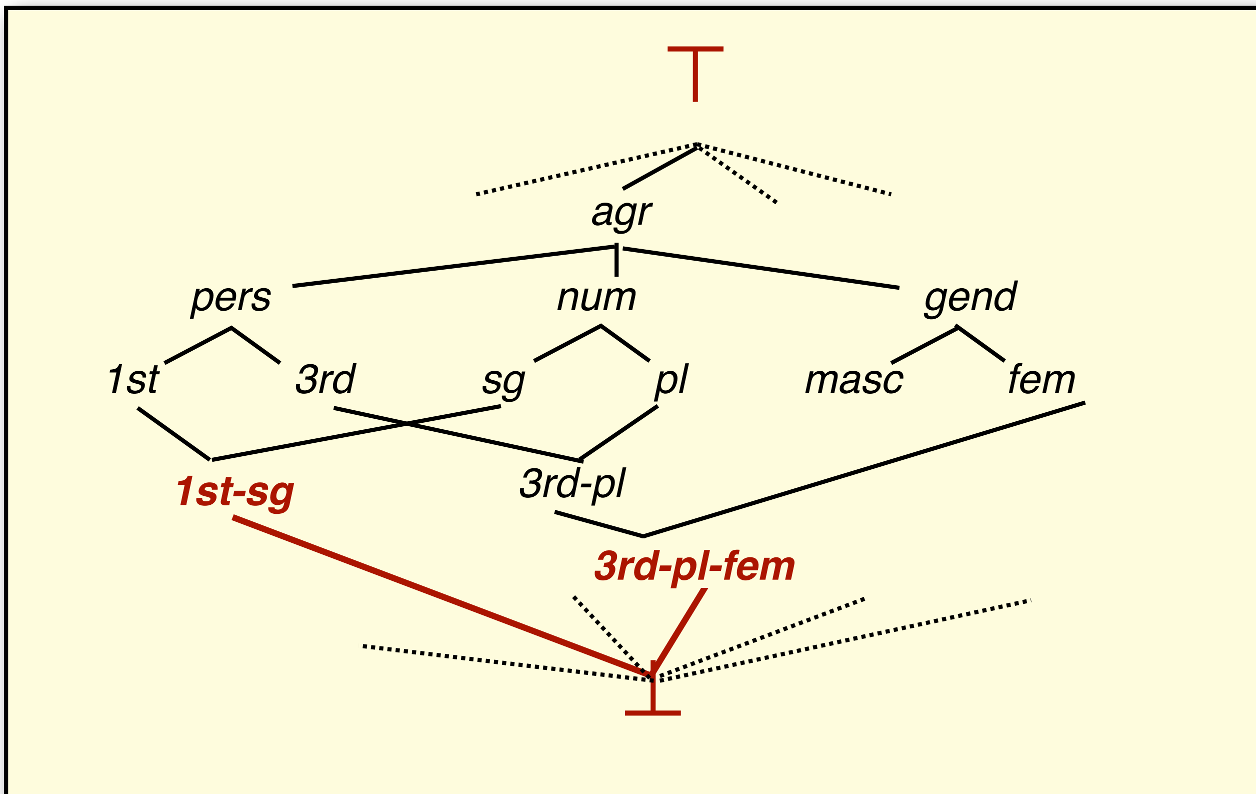
- each feature structure has a **type**
- each **type specifies which features** its structures can contain
- the **values of each feature are typed**
- types are arranged in a **multiple inheritance hierarchy:**

  ⊤('*top*') is the root*, pers* is a subtype of *agr, 3rd-pl-fem* is a subtype of *3rd-pl* and *fem* (and of *3rd, gend, pl, pers,…, agr)*

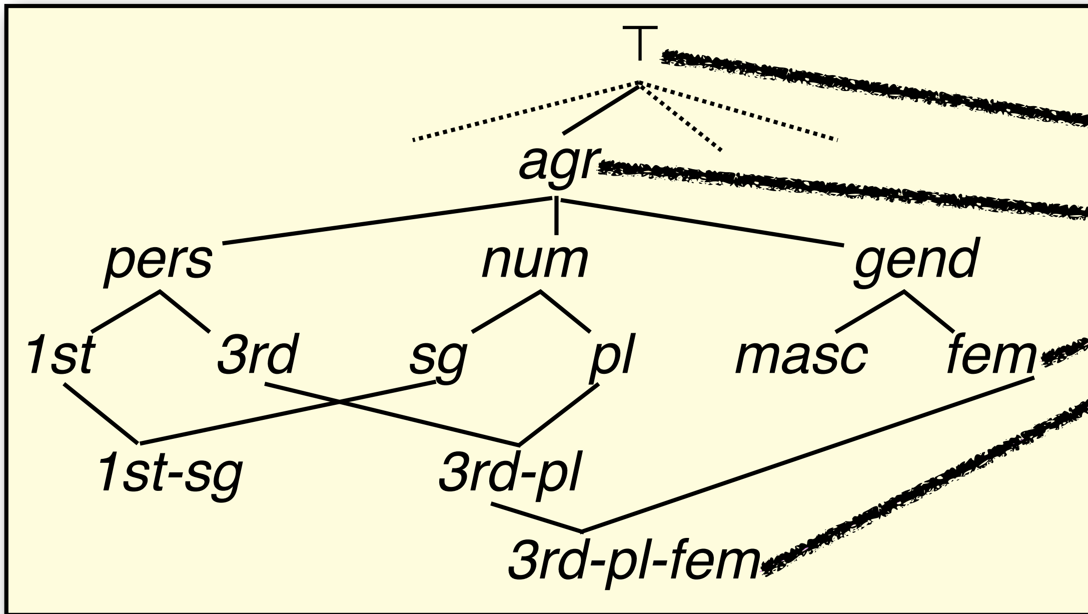


Another view of unification failure

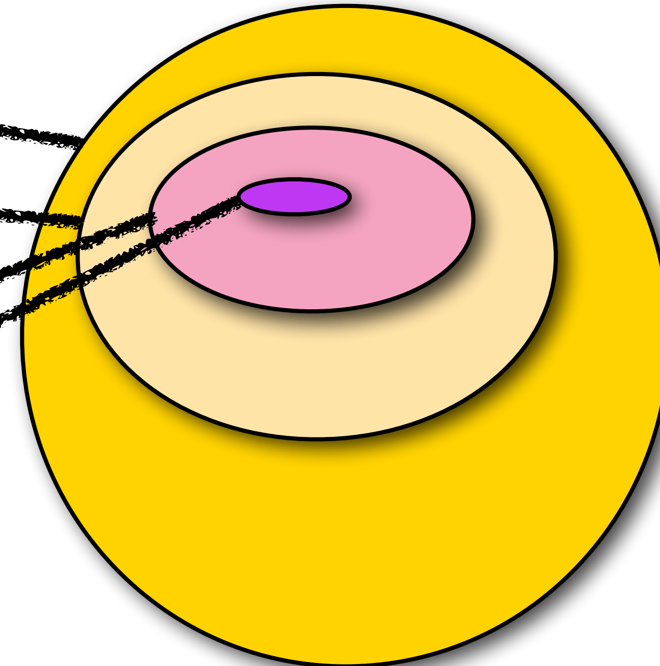
- ⊤ ('top') is the **root** of every type hierarchy
  (= the most general supertype)
- ⊥ ('bottom') is a **subtype** of every type.



Features as constraints



Type Hierarchy



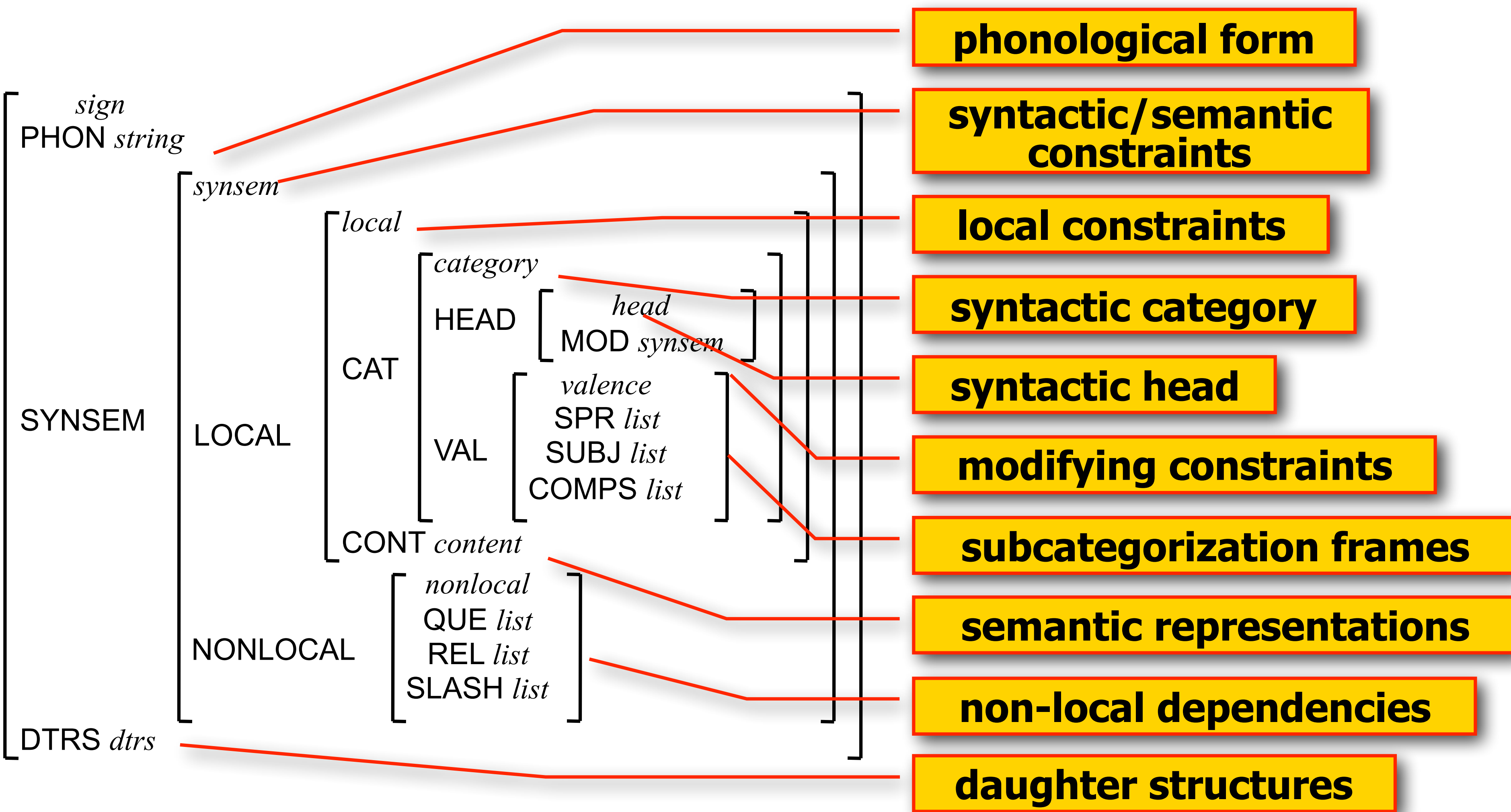
Universe of 'linguistic objects' (=strings)

Feature structure grammars

There are a number of grammar formalisms (the most widely used is Head-Driven Phrase Structure Grammar [HPSG, Pollard & Sag 1994]) that are based on this constraint-based view of feature structures.

(See next slide for an example)

HPSG signs (feature structures)



Feature structures: Summary

- We can use feature structures to refine or extend CFGs.
- Feature structures define constraints over linguistic objects (e.g. constituents)
- Feature structures may subsume each other.
- Feature structures can be simple or complex.
- A feature structure can be viewed as a directed graph.
- Feature structures can be combined via unification.
- Unification can be viewed as graph matching.
- Unification may fail.
- Feature structures may contain reentrancies (cycles).
- Feature structures can be typed.
- Types can be arrange in a type hierarchy.