Syntactic Theory

Lecture 25.01.2012

PD Dr.Valia Kordoni

Email: kordoni@coli.uni-sb.de

http://www.coli.uni-saarland.de/courses/syntactic-theory/2011/

Head-Driven Phrase Structure Grammar (HPSG)

Introduction – Part I -

History of HPSG and its influences

Head-Driven Phrase Structure Grammar—
Developed in the mid-1980s by Carl Pollard and Ivan Sag at Stanford University

- HPSG1: Pollard and Sag (1987)
  Formalism (typed features structures), subcategorization, LP rules, the hierarchical lexicon
- HPSG2: Pollard and Sag (1994) Chapters 1-8
  The structure of the sign, Control Theory, Binding Theory
- HPSG3: Pollard and Sag (1994) Chapter 9, "Reflections and Revisions"
  Valence features SUBJ, COMPS, SPR
- HPSG4, HPSG5, . . . ?
  Unbounded dependency constructions (Sag, 1997; Bouma, Malouf, & Sag, 1998), Linking Theory (Wechsler, 1995; Davis, 1997), semantic representation (Copestake, Flickinger, & Sag, 1997), argument realization (Sag & Miller, To appear)

History of HPSG and its influences (cont.)

Influenced by contemporary theories:

- Syntax
  - Generalized Phrase Structure Grammar (Gazdar, Klein, Pullum, & Sag, 1985)
  - Categorial Grammar (McGee Wood, 1993)
  - Lexical-Functional Grammar (Kaplan & Bresnan, 1982)
  - Construction Grammar (Goldberg, 1995)
  - Government-Binding Theory (Haegeman, 1994)
- Semantics
  - Situation Semantics (Barwise & Perry, 1983)
  - Discourse Representation Theory (Kamp & Reyle, 1993)

HPSG vs. “Classical” Phrase Structure Grammar

Similarities:

- Both are **monostratal**.
  Every sentence is associated with a single tree structure.
- Grammar rules have **local** scope only.
  Grammatical statements can only refer to a local tree (one node and its daughters). A tree is well-formed if and only if all its local trees are. (True of context-free PSG only)

HPSG vs. “Classical” Phrase Structure Grammar (cont.)

Differences:

- HPSG uses **complex categories** while Classical PSG uses simple/atomic ones.
- HPSG uses **Immediate Dominance (ID) schemas** and **Linear Precedence (LP)** rules instead of Classical PS rules.

  ID rules specify the mother and daughters in a local tree without specifying the order of the daughters. LP rules determine the relative order of the daughters in a local tree without making reference to the mother node.

  In addition, HPSG proposes several universal **principles** to further constrain the set of local trees admitted by the ID schemas.
- HPSG analyses include semantic representations in addition to syntactic representations.

HPSG vs. Transformational Grammar

Transformational Grammar—
"Chomskyan" frameworks, most recently formalized as Government and Binding Theory, Principles and Parameters, and Minimalism

Similarities:

- Both try to account for a similar range of data.
  Some analyses in HPSG (e.g., Binding Theory) are heavily influenced by earlier proposals in TG.

- Both are theories of **generative grammar**.
  Language is seen as the product of a system of rules and principles rather than a collection of strings to be described. The aim of both theories is to formulate these general principles. Furthermore, the generalizations are meant to say something about human linguistic **knowledge**.

HPSG vs. Transformational Grammar (cont.)

Differences:

- HPSG is **non-derivational**, TG is derivational.
  TG analyses start with a base generated tree, which is then subject to a variety of transformations (e.g., movement, deletion, reanalysis) that produce the desired surface structure. HPSG analyses generate only the surface tree. Rule ordering is impossible in HPSG because there is no notion of sequential derivation.
- HPSG constraints are **local**, TG allows non-local statements.
- **Complex categories** in HPSG are more complex than in TG.
  TG uses atomic categories carrying binary feature specifications. HPSG categories are very elaborate in comparison.
- HPSG is more committed to precise **formalization** than TG.
- HPSG is better suited to **computational implementation** than TG.

Key properties and their consequences

1. HPSG is monostratal, declarative, non-derivational.
   No transformations, no rule ordering. In addition, analyses are surface-oriented, with a desire to avoid abstract structures such as traces and functional categories.

2. HPSG is constraint-based.
   A structure is well-formed if and only if it satisfies all relevant constraints. Constraints are not violable (as in Optimality Theory, for example). Furthermore, constraints apply only to local trees (although local constraints can interact to have non-local effects).

3. HPSG is a lexicalist theory.
   Strong Lexicalism (Scalise, 1984). Word-internal structure (e.g., morphology) and phrase structure are handled separately. Phrasal rules cannot manipulate sub-parts of words. Lexically-governed processes such as valence alternation must be analyzed lexically.

Key properties and their consequences (cont.)

4. HPSG is "head-driven."
   (More on this later)

5. HPSG is a "unification-based" framework where all linguistic objects are represented as "typed feature structures."
   (Next section)

Psycholinguistic evidence

HPSG aims to model our knowledge of language. Support for the model proposed by HPSG comes from the fact that it accommodates several empirical facts about human language processing:

- Human language processing is **incremental**:
  Partial interpretations can be generated for partial utterances.

  HPSG constraints can apply to partial structures as well as complete trees.

- HLP is **integrative**:
  Linguistic interpretations depend on a large amount of non-linguistic information (e.g., world knowledge).

  The signs used in HPSG (typed feature structures) can incorporate both linguistic and non-linguistic information using the same formal representation.

Psycholinguistic evidence (cont.)

- HLP is **order-independent**:
  There is no fixed sequence in which pieces of information are consulted and incorporated into a linguistic interpretation.

  HPSG is a declarative model, so information can be added in any order.

- HLP is **reversible**:
  We can produce and understand the same kinds of utterances.

  The grammar of HPSG is process-neutral. It can be used for either production or comprehension.

Feature Structures and Feature Structure Descriptions

Feature structure—

- Provides a description/representation of an object (linguistic or non-linguistic) by specifying information about its **attributes**
- Formally, it is a **directed acyclic graph** (DAG): a root node, directed edges corresponding to attributes, and end nodes corresponding to values. ("acyclic": no circular paths)

Feature Structures and Feature Structure Descriptions (cont.)

- More conveniently decribed using an **attribute-value matrix** (AVM):

(1)

$$\begin{bmatrix} \text{TITLE} & \textit{HPSG} \\ \text{AUTHOR1} & \textit{Pollard} \\ \text{AUTHOR2} & \textit{Sag} \\ \text{COVER} & \textit{blue} \\ \text{PAGES} & 440 \end{bmatrix}$$

Each row is an attribute-value pair, or **feature**. The order of the rows is unimportant.

Feature Structures and Feature Structure Descriptions (cont.)

- Within a single AVM, an attribute can only take one value. The following is an improper AVM; it does not describe any feature structure:

(2)

$$\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{SEX} & \textit{male} \\ \text{AGE} & 29 \\ \text{SEX} & \textit{female} \end{bmatrix}$$

It is common practice to refer to AVMs as "feature structures" although strictly speaking they are feature structure *descriptions* (FSDs).

Recursive Structure

The value of an attribute is not necessarily atomic; it can specify attributes of its own. In other words, in an AVM, the value of an attribute can be another AVM:

(3)

$$\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & 29 \\ \text{ADDRESS} & \begin{bmatrix} \text{NUMBER} & 10 \\ \text{STREET} & \textit{Main} \\ \text{CITY} & \textit{Springfield} \end{bmatrix} \\ \text{FAVORITES} & \begin{bmatrix} \text{FOOD} & \textit{cheese} \\ \text{COLOR} & \textit{blue} \end{bmatrix} \end{bmatrix}$$

Recursive Structure (cont.)

In fact, the value of an attribute is *always* an AVM. Atomic values should be thought of as very simple AVMs that have no attributes.

In DAG terms: The end node of an attribute edge can itself be the root node of a feature structure, with attribute edges leaving from it. The edges leading from the root node of the entire DAG to an atomic value make up an "attribute path." E.g., the value of the path "ADDRESS | CITY" is *Springfield*.

An atomic value is a minimal DAG consisting of a single labelled node with no outward edges.

Structure sharing

Two attributes can share the same value. Two cases:

- **Type identity**
  The values are the same by accident, but in fact they are independently specified. There are really two values that happen to look the same. They could "just as easily" have been different.

- **Token identity**
  Some principle (physical law, fact about the world, grammatical rule) insists that the two attributes have identical values. There is only one value, and both attributes share it.

Structure sharing (cont.)

Example: Kim and Sandy are twins.

(4)

$$\begin{bmatrix} \text{TWIN1} & \begin{bmatrix} \text{NAME} & \textit{Kim} \\ \text{AGE} & \boxed{1}\ 29 \\ \text{FAVORITES} & \begin{bmatrix} \text{COLOR} & \textit{blue} \end{bmatrix} \end{bmatrix} \\ \text{TWIN2} & \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & \boxed{1}\ 29 \\ \text{FAVORITES} & \begin{bmatrix} \text{COLOR} & \textit{blue} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Structure sharing (cont.)

It is a fact about the world that twins have the same age, so the values for AGE are token identical, or "structure-shared." This is notated with matching boxed indices. On the other hand, the fact that Kim and Sandy have the same favorite color is accidental; these values are not structure-shared.

In DAG terms—

- Type identity: The two attribute edges have distinct end nodes.
- Token identity: The two attribute edges point to the same end node (structure-sharing).

Structure sharing (cont.)

Structure sharing is not allowed to result in circular or cyclical paths:

(5)

$$\boxed{1}\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{SIBLING} & \begin{bmatrix} \text{NAME} & \textit{Kim} \\ \text{SIBLING} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

This is not a well-formed feature structure description because is does not describe a DAG (directed acyclic graph).

Partial descriptions

Feature structures (DAGs) are assumed to represent an object as fully as possible, by specifying all values for all attributes.

Feature structure descriptions (AVMs), on the other hand, can be **partial** descriptions, specifying some features and omitting others. An underspecified AVM can correspond to a set of feature structures. As the AVM becomes more fully specified, the set of DAGs it describes becomes smaller and smaller, as fewer and fewer of them are compatible with the information given in the AVM.

Subsumption

The fact that feature structure descriptions can be more or less informative allows us to define a partial ordering on the set of FSDs.

For two AVMs A and B, A **subsumes** B ($A \preceq B$) iff

- B is at least as informative as A. That is, all the attributes and values specified in A are also found in B. B might specify other features as well, but it must include all of the ones in A.
- The set of feature structures described by A is a superset of the set of feature structures described by B.

Subsumption (cont.)

Formal definition of subsumption:

(6) a. For atomic values a and b (remember these are considered to be AVMs), $a \preceq b \Leftrightarrow b \preceq a \Leftrightarrow a = b$.[1]

b. For non-atomic AVMs A and B, A $\preceq$ B iff

i. for every attribute path in A, the same path exists in B and its value in A subsumes its value in B

ii. for every pair of paths that is structure-sharing in A, the same pair of paths is structure-sharing in B.

[1] **NB**: This will change once we introduce *sorts*.

Subsumption (cont.)

(7)

$$\underbrace{\begin{bmatrix} \text{AGE} & \boxed{1}\ 29 \\ \text{ADDRESS} & \begin{bmatrix} \text{STREET} & \textit{Main} \end{bmatrix} \\ \text{SIBLING} & \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & \boxed{1} \end{bmatrix} \end{bmatrix}}_{\text{A}} \preceq \underbrace{\begin{bmatrix} \text{NAME} & \textit{Kim} \\ \text{AGE} & \boxed{1}\ 29 \\ \text{ADDRESS} & \begin{bmatrix} \text{STREET} & \textit{Main} \\ \text{CITY} & \textit{Springfield} \end{bmatrix} \\ \text{SIBLING} & \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & \boxed{1} \end{bmatrix} \end{bmatrix}}_{\text{B}}$$

Subsumption (cont.)

Notes:

- Subsumption is
  - reflexive: $A \preceq A$
  - transitive: if $A \preceq B$ and $B \preceq C$, then $A \preceq C$
  - anti-symmetric: if $A \preceq B$ and $B \preceq A$, then $A = B$.[2]

Subsumption (cont.)

- For two AVMs A and B, it is possible that neither one subsumes the other:

(8) a.

$$\underset{\text{A}}{\begin{bmatrix} \text{NAME} & \textit{Kim} \\ \text{AGE} & 29 \end{bmatrix}} \qquad \underset{\text{B}}{\begin{bmatrix} \text{AGE} & 30 \end{bmatrix}}$$

b.

$$\underset{\text{A}}{\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{JOB} & \textit{photographer} \end{bmatrix}} \qquad \underset{\text{B}}{\begin{bmatrix} \text{AGE} & 29 \\ \text{NATIONALITY} & \textit{British} \end{bmatrix}}$$

Subsumption (cont.)

- The minimum element with respect to the subsumption ordering is the feature structure that specifies no information at all (no attributes, no values). It is called "top" and is written $\top$ or [ ]. Top subsumes every other AVM, because every other AVM contains as least as much information as top: $\top \preceq A$ for all A.

Unification

Since AVMs can be partial descriptions, we might have several different descriptions of the same feature structure. It must be the case, though, that if two AVMs A and B are descriptions of the same object (or set of objects), then the information in A and B must be compatible. In particular, A and B cannot have different values for the same attributes or attribute paths.

[2] A and B are type identical, not necessarily token identical.

Unification (cont.)

(9) a.

$$\begin{bmatrix} \text{AGE} & 29 \\ \text{JOB} & \textit{photographer} \end{bmatrix} \quad \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & 29 \end{bmatrix}$$

A B

b.

$$\begin{bmatrix} \text{AGE} & 29 \\ \text{JOB} & \textit{photographer} \end{bmatrix} \quad \begin{bmatrix} \text{JOB} & \textit{photographer} \\ \text{AGE} & 30 \end{bmatrix}$$

A B

Unification (cont.)

In the case where A and B contain compatible, consistent information, we say that the **unification** of A and B exists. The unification of A and B in (9a) is:

(10)

$$\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & 29 \\ \text{JOB} & \textit{photographer} \end{bmatrix}$$

The unification of two feature structures contains all the information from both feature structures put together, but nothing more than that.

Unification (cont.)

Definition:

(11) For AVMs A and B, the unification C of A and B ($C = A \wedge B$) is the least informative AVM that is subsumed by both A and B.

(Even more formally: C is the least upper bound of A and B with respect to the subsumption ordering $\preceq$.)

Notes:

- The unification c of two atomic AVMs a and b is defined if and only if $a = b$ and in that case $c = a \wedge b = a = b$.[3]
- Unification works recursively on embedded AVMs.
- Unification preserves structure-sharing.

[3] **NB**: This will also change after we introduce sorts. (see fn. 1)

Unification (cont.)

(12)

$$\begin{bmatrix} \text{NAME} & \textit{Kim} \\ \text{ADDRESS} & \begin{bmatrix} \text{CITY} & \textit{Springfield} \end{bmatrix} \\ \text{SIBLING} & \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & 29 \end{bmatrix} \end{bmatrix} \quad \text{A}$$

$$\wedge$$

$$\begin{bmatrix} \text{AGE} & \boxed{1} \\ \text{ADDRESS} & \begin{bmatrix} \text{STREET} & \textit{Main} \end{bmatrix} \\ \text{SIBLING} & \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & \boxed{1} \end{bmatrix} \end{bmatrix} \quad \text{B}$$

$$=$$

$$\begin{bmatrix} \text{NAME} & \textit{Kim} \\ \text{AGE} & \boxed{1}\ 29 \\ \text{ADDRESS} & \begin{bmatrix} \text{STREET} & \textit{Main} \\ \text{CITY} & \textit{Springfield} \end{bmatrix} \\ \text{SIBLING} & \begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & \boxed{1} \end{bmatrix} \end{bmatrix} \quad \text{C}$$

Unification (cont.)

If two AVMs A and B contain inconsistent feature specifications, then they "fail to unify," or their unification does not exist. Alternatively, we can write

(13) $\mathrm{A} \wedge \mathrm{B} = \bot$

where $\bot$ ("bottom") is an "improper" AVM that cannot describe any object (the opposite of $\top$, which can describe any object).

Unification (cont.)

As in the case of structure sharing, unification does not succeed if it results in cyclical structure:

(14)

$$\underbrace{\begin{bmatrix} \text{ATTRIBUTE1} & \begin{bmatrix} \text{ATTRIBUTE3} & \boxed{1} \end{bmatrix} \\ \text{ATTRIBUTE2} & \boxed{1} \end{bmatrix}}_{\text{A}} \wedge \underbrace{\begin{bmatrix} \text{ATTRIBUTE1} & \boxed{2} \\ \text{ATTRIBUTE2} & \begin{bmatrix} \text{ATTRIBUTE3} & \boxed{2} \end{bmatrix} \end{bmatrix}}_{\text{B}}$$

$$= \underbrace{\begin{bmatrix} \text{ATTRIBUTE1} & \boxed{2}\begin{bmatrix} \text{ATTRIBUTE3} & \boxed{1} \end{bmatrix} \\ \text{ATTRIBUTE2} & \boxed{1}\begin{bmatrix} \text{ATTRIBUTE3} & \boxed{2} \end{bmatrix} \end{bmatrix}}_{\text{C}} = \bot$$

Unification (cont.)

Properties of unification (from Pollard and Sag (1987), p. 38):

- idempotency: $A \wedge A = A$
- commutativity: $A \wedge B = B \wedge A$
- associativity: $(A \wedge B) \wedge C = A \wedge (B \wedge C)$
- $\top \wedge A = A$
- $\bot \wedge A = \bot$
- link between subsumption and unification: $A \preceq B \Leftrightarrow A \wedge B = B$

References

Barwise, J., & Perry, J. (1983). *Situations and attitudes.* Cambridge, MA: MIT Press.

Bouma, G., Malouf, R., & Sag, I. (1998). A unified theory of complement, adjunct, and subject extraction. In G. Bouma, G.-J. M. Kruijff, & R. T. Oehrle (Eds.), *Proceedings of the joint conference on Formal Grammar, Head-Driven Phrase Structure Grammar, and Categorial Grammar* (pp. 83–97). Saarbrücken: .

Copestake, A., Flickinger, D., & Sag, I. A. (1997). *Minimal Recursion Semantics: An introduction.*

Davis, A. (1997). *Lexical semantics and linking in the hierarchical lexicon.* Unpublished doctoral dissertation, Stanford University.

Gazdar, G., Klein, E., Pullum, G. K., & Sag, I. A. (1985). *Generalized Phrase Structure Grammar.* Cambridge, MA: Harvard University Press.

References (cont.)

Goldberg, A. E. (1995). *Constructions: A Construction Grammar approach to argument structure.* Chicago: University of Chicago Press.

Haegeman, L. (1994). *Introduction to Government and Binding Theory* (2nd ed.). Oxford: Blackwell.

Kamp, H., & Reyle, U. (1993). *From discourse to logic.* Dordrecht: Kluwer.

Kaplan, R., & Bresnan, J. (1982). Lexical-Functional Grammar: A formal system for grammatical representation. In J. Bresnan (Ed.), *The mental representation of grammatical relations* (p.). Cambridge, MA: MIT Press.

McGee Wood, M. (1993). *Categorial grammars.* London: Routledge.

Pollard, C., & Sag, I. A. (1987). *Information-based syntax and semantics.* Stanford: CSLI.

References (cont.)

Sag, I. A. (1997). English relative clause constructions. *Journal of Linguistics*, *33*, 431–484.

Sag, I. A., & Miller, P. H. (To appear). French clitic movement without clitics or movement. *Natural Language and Linguistic Theory*.

Scalise, S. (1984). *Generative morphology*. Dordrecht: Foris.

Wechsler, S. (1995). *The semantic basis of argument structure*. Stanford: CSLI.

Head-Driven Phrase Structure Grammar (HPSG)

Introduction – Part II -

Changes to Part I

1. Following Carpenter (1992), we will use $\sqsubseteq$ (not $\preceq$) to represent the subsumption relation. Pollard and Sag (1987) use $\preceq$ for the *extension* relation, which is dual to subsumption:

   (1) A subsumes B if and only if B extends A
   $A \sqsubseteq B \Leftrightarrow B \preceq A$

   Carpenter (1992) also reverses the roles of $\top$ and $\bot$ in Pollard and Sag (1987). We will try to avoid confusion by using [ ] exclusively to denote the least informative/most general feature structure.

2. Feature structures are assumed to be acyclic in HPSG1, as in PATR-II (Pereira & Shieber, 1984). This requirement is dropped in more recent versions of HPSG.

   (2) From Engdahl and Vallduví (1996), p. 11:

$$\boxed{1}\ \textit{word}\left[\begin{array}{ll}\text{PHON} \mid \text{ACCENT} & \text{A} \\ \text{INFO-STRUCT} \mid \text{FOCUS} & \boxed{1}\end{array}\right]$$

Typed Feature Structures

With the feature structures we have so far, there are no restrictions on the attributes we can assign to an object, or the values that those attributes can take. We have imposed our own world knowledge and common sense to come up with reasonable descriptions:

(3)

$$\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & 29 \\ \text{ADDRESS} & \begin{bmatrix} \text{NUMBER} & 10 \\ \text{STREET} & \textit{Main} \\ \text{CITY} & \textit{Springfield} \end{bmatrix} \end{bmatrix}$$

Typed Feature Structures (cont.)

There is nothing in the formalism, however, that disallows feature structures described by the following AVM:

(4)

$$\begin{bmatrix} \text{NAME} & \textit{Sandy} \\ \text{AGE} & \begin{bmatrix} \text{NUMBER} & 10 \\ \text{STREET} & \textit{Main} \\ \text{CITY} & \textit{Springfield} \end{bmatrix} \\ \text{FONT} & \textit{Helvetica} \end{bmatrix}$$

The feature [NAME : *Sandy*] is the only appropriate piece of information here. The attribute AGE could be appropriate, but it takes the wrong kind of value here. And FONT is probably inappropriate for a person, whatever its value.

What attributes/values are appropriate? It depends on what kind of object the feature structure is meant to model. We need to extend our formalism to include this information.

Typed Feature Structures (cont.)

Typed feature structure—
A feature structure (a directed graph with labelled edges) where *every* node is labelled with a **sort symbol** corresponding to the ontological category of the object being modelled by that node (and its outward edges).

Typed feature structure descriptions (AVMs) will also include this type information:

(5)

$$\textit{person}\left[\begin{array}{ll} \textsc{name} & \textit{Sandy} \\ \textsc{age} & 29 \\ \textsc{address} & \textit{address}\left[\begin{array}{ll} \textsc{number} & 10 \\ \textsc{street} & \textit{Main} \\ \textsc{city} & \textit{Springfield} \end{array}\right] \end{array}\right]$$

The Type hierarchy

The set of types models the set of ontological categories in the domain of objects being described. Every type specifies the set of attributes that is appropriate for it, and the types of the attributes' values. For example, a type *garment* representing an item of clothing might introduce the following attributes and values:

(6)

$$garment \begin{bmatrix} \text{MATERIAL} & fabric \\ \text{COLOR} & color \end{bmatrix}$$

The Type hierarchy (cont.)

Particular items of clothing might introduce different features

(7) a.

$$\textit{shirt}\left[\text{NECK}\quad \textit{measurement}\right]$$

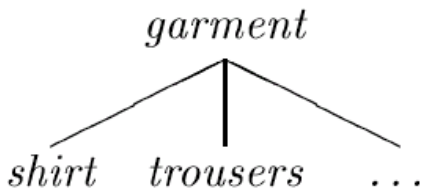
b.

$$\textit{trousers}\left[\text{WAIST}\quad \textit{measurement}\right]$$

Inheritance

Furthermore, types are organized into a subsumption hierarchy, with more general supertypes classified into more specific subtypes:

(8)



Here, *garment* subsumes both *shirt* and *trousers*—i.e., the subtypes must contain all the information in the supertype *garment*. In particular, they both **inherit** the attribute and value specifications in (6):

(9)

$$\underset{shirt}{\begin{bmatrix} \text{MATERIAL} & fabric \\ \text{COLOR} & color \\ \text{NECK} & length \end{bmatrix}} \qquad \underset{trousers}{\begin{bmatrix} \text{MATERIAL} & fabric \\ \text{COLOR} & color \\ \text{WAIST} & length \end{bmatrix}}$$

Type declaration

Components of a (basic) type hierarchy:

1. A root type, which is a supertype of all other types (in other words, [ ]).
2. Every type, starting with the root type, is **partitioned** into a set of disjoint, exhaustive subtypes.
3. Every type is defined by specifying the features (attribute-value pairs) it selects.
4. Subtypes inherit the specifications of all their supertypes. They can introduce more specific feature information or introduce altogether new attributes.

Subsumption and unification

Atomic types—
All attributes take typed AVMs as values. At the ends of attribute paths, we find typed AVMs that specify no features. The only information these AVMs contain is type information. For example, in (5) above, the path [ADDRESS | CITY : *Springfield*] ends in an atomic type. The path could be written:

(10)

$$\left[\text{ADDRESS} \mid \text{CITY} \quad \textit{springfield}\left[\ \right]\right]$$

Subsumption and unification (cont.)

Subsumption of typed feature structures is still determined by information content: a less informative AVM subsumes a more informative one.

(11) For typed AVMs A of type σ and B of type τ, A $\sqsubseteq$ B iff

a. $\sigma \sqsubseteq \tau$ (i.e., τ is a subtype of σ)

b. for every feature [F : A′] in A, there is a feature [F : B′] in B, and A′ $\sqsubseteq$ B′, where A′ and B′ are (possibly atomic) typed AVMs.

Unification is also defined as before: the combination of information from two AVMs, as long as it does not result in inconsistency. Since we are assuming pairwise disjoint type partitions, two types can only give consistent information if one is a subtype of the other (or if they are identical types). And in that case, we can apply the subsumption-unification rule from Handout 1:

(12) $\sigma \sqsubseteq \tau \Leftrightarrow \sigma \wedge \tau = \tau$

That is, the unification of the two types is identical to the subtype, or more informative type.

Subsumption and unification (cont.)

(13) a.

$$_{shirt}\begin{bmatrix}\text{MATERIAL} & silk\\ \text{COLOR} & color\\ \text{NECK} & 15.5\end{bmatrix} \wedge {}_{trousers}\begin{bmatrix}\text{MATERIAL} & wool\\ \text{COLOR} & black\\ \text{WAIST} & 32\end{bmatrix} = \text{undef.}$$

b.

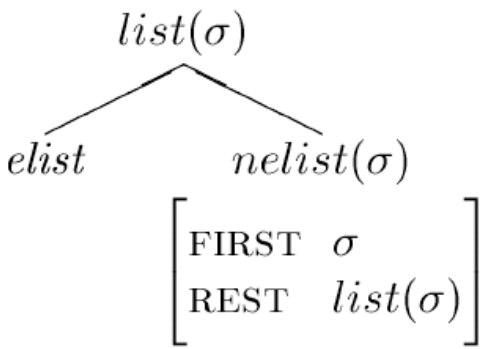
$$_{garment}\begin{bmatrix}\text{MATERIAL} & cotton\\ \text{COLOR} & green\end{bmatrix} \wedge {}_{trousers}\begin{bmatrix}\text{MATERIAL} & fabric\\ \text{WAIST} & length\end{bmatrix}$$

$$= {}_{trousers}\begin{bmatrix}\text{MATERIAL} & cotton\\ \text{COLOR} & green\\ \text{WAIST} & length\end{bmatrix}$$

Lists and Sets

Ordered lists are described using the parametrized sort $list(\sigma)$, where σ is a sort symbol. $list(\sigma)$ is partitioned into subsorts *empty-list* and *nonempty-list*(σ):

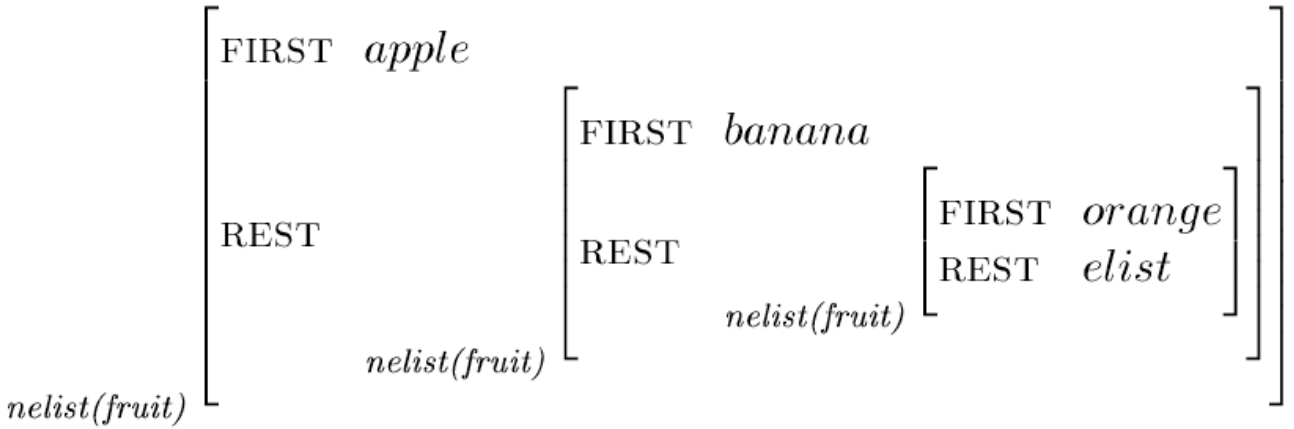
(14)



Lists and Sets (cont.)

For example, the following AVM represents a list of three subtypes of *fruit*:

(15)



However this is more commonly abbreviated with the following list notation:

(16) $\langle apple, banana, orange \rangle$

Lists and Sets (cont.)

The type $set(\sigma)$ is used to describe sets of feature structures of type σ. We use familiar set notation for values of this type:

(17) $\{apple, banana, orange\}$

It should be noted that both of these types introduce some technical complications (see Carpenter (1992), pp. 34, 78–79, for discussion). For our purposes, however, an intuitive understanding of lists and sets will suffice.

Full Specification

Feature structures in HPSG are assumed to be maximally informative in two ways. If we have a feature structure of type σ:

1. It must be **totally well-typed**.
   All of the attributes that are appropriate for σ according to the type hierarchy must actually appear in the feature structure, with the appropriately typed values.

2. It must be **sort-resolved**.
   σ cannot have any subtypes. It must be the most specific type available in the hierarchy for labelling the feature structure.

Note, however, that these two requirements apply to feature structures, and not to AVMs. The fact that AVMs can be left partially specified is essential to HPSG analyses.

The structure of the sign in HPSG

The appendix of P&S94 includes a statement of the type hierarchy assumed in Chapters 1-8 ("HPSG2").

The basic sort in HPSG is *sign*, used to describe lexical items (of type *word*) and phrases (of type *phrase*). All signs carry the following two features:

- [PHONOLOGY : *list*(*phoneme-string*)]
  Encodes the phonological representation of the sign. A neglected part of the theory.
- [SYNSEM : *synsem*]
  Syntax and semantics, loosely speaking (see next section)

Lexical signs also have a MORPHOLOGY attribute, whose value gives information about the word's sublexical structure: its stem and affixes. Phrasal signs have a DAUGHTERS attribute, which encodes constituent structure.

The structure of the sign in HPSG (cont.)

The type *synsem* introduces the attributes LOCAL and NON-LOCAL. We will study NON-LOCAL features in connection with unbounded dependencies (e.g., *wh*-movement). For now we will focus on two features on the LOCAL path which contain more familiar kinds of grammatical information:

(18)

$$\mathit{local}\begin{bmatrix}\textsc{category} & \mathit{category}\\ \textsc{content} & \mathit{content}\\ \textsc{context} & \mathit{context}\end{bmatrix}$$

Syntactic Category and Valence

The value of CATEGORY encodes information about (1) the sign's syntactic category ("part of speech") and (2) its syntactic valence (i.e., its potential to combine with other signs to form larger phrases).

Part of speech information is given via the feature [HEAD : *head*], where *head* is the supertype for the familiar labels: *noun, verb, adjective, preposition, determiner, marker*. Each of these subtypes selects a particular set of **head features**, which are the subject of one of the fundamental constraints in HPSG, the Head Feature Principle (HFP).

Syntactic Category and Valence (cont.)

Valence information is represented by three list-valued attributes:

(19)

$$\begin{bmatrix} \text{SYNSEM} \mid \text{LOC} \mid \text{CAT} \mid \text{VALENCE} & \begin{bmatrix} \text{SUBJECT} & list(synsem) \\ \text{SPECIFIER} & list(synsem) \\ \text{COMPLEMENTS} & list(synsem) \end{bmatrix} \end{bmatrix}$$

If any of these lists are specified as non-empty, the sign has the potential to combine with another sign (like a functor category in Categorial Grammar). A sign with a non-empty valence list is called "unsaturated."

Semantic representation

Information about truth-conditional meaning is found in the CONTENT value. The structure of this value depends very much on the subsort of *content* that labels it:

- *nominal-object*
  An individual/entity (or a set of them), associated with a referring index, bearing agreement features
- *parameterized-state-of-affairs*
  A partial situation (as in Situation Semantics): an event relation along with role names for identifying the participants in the event
- *quantifier*
  some, all, every, a, the, ...

The structure of CONTENT values has been reformulated as "Minimal Recursion Semantics" (Copestake, Flickinger, & Sag, 1997), which allows underspecification of quantifier scope and has computational advantages. For the most part, though, we will follow HPSG2.

References

Carpenter, B. (1992). *The logic of typed feature structures.* Cambridge: Cambridge University Press.

Copestake, A., Flickinger, D., & Sag, I. A. (1997). *Minimal Recursion Semantics: An introduction.*

Engdahl, E., & Vallduví, E. (1996). Information packaging in HPSG. In C. Grover & E. Vallduví (Eds.), *Edinburgh Working Papers in Cognitive Science, volume 12: Studies in HPSG* (pp. 1–32). Edinburgh: Centre for Cognitive Science.

Pereira, F. C., & Shieber, S. M. (1984). The semantics of grammar formalisms seen as computer languages. In *Proceedings of the 10th international conference on computational linguistics* (pp. 123–129). Stanford.

Pollard, C., & Sag, I. A. (1987). *Information-based syntax and semantics.* Stanford: CSLI.

Head-Driven Phrase Structure Grammar (HPSG)

Introduction – Part III -

The building blocks of HPSG grammars

In HPSG, sentences, words, phrases, and multisentence discourses are all represented as **signs** = complexes of phonological, syntactic/semantic, and discourse information.

We can (and will) view HPSG grammars in two different ways:

1. From a linguistic perspective
2. From a formal perspective

HPSG grammars from a linguistic perspective

From a linguistic perspective, an HPSG grammar consists of

a) a lexicon
   licensing basic words

b) lexical rules
   licensing derived words

c) immediate dominance (ID) schemata
   licensing constituent structure

d) linear precedence (LP) statements
   constraining word order

e) a set of grammatical principles
   expressing generalizations about linguistic objects

HPSG feature structures

HPSG is nonderivational, but in some sense, HPSG has several different levels (layers of features)

A feature structure is a directed acyclic graph (DAG), with arcs representing features going between values

Each of these feature values is itself a complex object:

- The **type** *sign* has the **features** PHON and SYNSEM appropriate for it
- The feature SYNSEM has a **value** of type *synsem*
- This type itself has relevant features (LOCAL and NON-LOCAL)

HPSG grammars from a formal perspective

From a formal perspective, an HPSG grammar consists of

- **the signature** as declaration of the domain, and
- **the theory** constraining the domain.

The signature

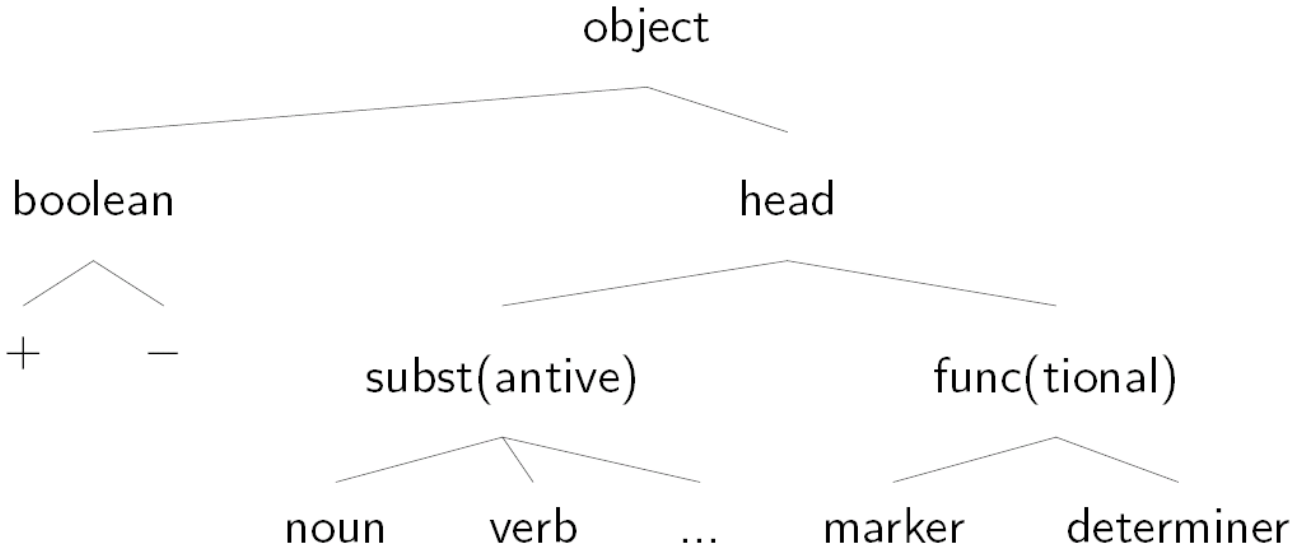
- defines the ontology ('declaration of what exists'):
  - which kind of objects are distinguished, and
  - which properties of which objects are modeled.

- consists of
  - the type (or sort) hierarchy and
  - the appropriateness conditions, defining which type has which appropriate attributes (or features) with which appropriate values.

    Some **atomic** types have no feature appropriate for them

Example excerpt of a signature

Here, we leave out the appropriateness conditions and just show a hierarchy of types



Sort-resolved

Based on the example signature, the following two descriptions are equivalent:

(1) a. *func*
b. *marker* $\vee$ *determiner*

That is, a type (or sort) is really a disjunction of its **maximally specific subtypes**

Models of linguistic objects

- As mentioned, the objects are modelled by feature structures, which are depicted as directed graphs.

- Since these models represent objects in the world (and not knowledge about the world), they are total with respect to the ontology declared in the signature. Technically, one says that these feature structures are
  - *totally well-typed*: Every node has all the attributes appropriate for its type and each attributes has an appropriate value.
  - *sort-resolved*: Every node is of a maximally specific type.

Structure sharing

The main explanatory mechanism in HPSG is that of **structure-sharing**, equating two features as having the exact same value (token-identical)

$$\begin{bmatrix} \textit{word} & \\ \textsc{phon} & \langle \textit{walks} \rangle \\ \textsc{synsem}|\textsc{loc} & \begin{bmatrix} \textsc{cat}|\textsc{subcat} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{3rd,sing}] \right\rangle \\ \textsc{content} & \begin{bmatrix} \textit{laugh'} \\ \textsc{laugher} \ \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The index of the NP on the SUBCAT list is said to **unify** with the value of LAUGHER

Descriptions

A **description language** and its abbreviating **attribute-value matrix (AVM) notation** is used to talk about sets of objects. Descriptions consists of three building blocks:

- **Type** decriptions single out all objects of a particular type, e.g., *word*
- **Attribute-value pairs** describe objects that have a particular property. The attribute must be appropriate for the particular type of object, and the value can be any kind of description, e.g., $\left[\textsc{spouse}\ \left[\textsc{name}\ \textit{mary}\right]\right]$
- **Tags** (structure sharing) to specify **token identity**, e.g. $\boxed{1}$

Descriptions (cont.)

Complex descriptions are obtained by combining descriptions with the help of conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$). In the AVM notation, conjunction is implicit.

A **theory** (in the formal sense) is a set of description language statements, often referred to as the constraints.

- The theory singles out a subset of the objects declared in the signature, namely those which are grammatical.
- A linguistic object is admissible with respect to a theory iff it satisfies each of the descriptions in the theory and so does each of its substructures.

Description example

A verb, for example, can specify that its subject be masculine singular (as Russian past tense verbs do):

(2) a. Ya spal.
I$_{masc.sg}$ slept$_{masc.sg}$
b. On spal.
He$_{masc.sg}$ slept$_{masc.sg}$

(3)
$$\begin{bmatrix} \textit{word} \\ \textsc{synsem}|\textsc{loc} \begin{bmatrix} \textsc{cat}|\textsc{head} \ \textit{noun} \\ \textsc{content} \begin{bmatrix} \textsc{index} \begin{bmatrix} \textsc{num} \ \textit{sing} \\ \textsc{gen} \ \textit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

This doesn't specify the entire (totally well-typed) feature structure, just what needs to be true in the feature structure.

Subsumption

The description in (3) is said to **subsume** both of the following (partial) feature structures:

(4) a.
$$\begin{bmatrix} \textit{word} \\ \textsc{synsem}|\textsc{loc} \begin{bmatrix} \textsc{cat}|\textsc{head} & \textit{noun} \\ \textsc{content} & \begin{bmatrix} \textsc{index} & \begin{bmatrix} \textsc{per} & \textit{1st} \\ \textsc{num} & \textit{sing} \\ \textsc{gen} & \textit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{word} \\ \textsc{synsem}|\textsc{loc} \begin{bmatrix} \textsc{cat}|\textsc{head} & \textit{noun} \\ \textsc{content} & \begin{bmatrix} \textsc{index} & \begin{bmatrix} \textsc{per} & \textit{3rd} \\ \textsc{num} & \textit{sing} \\ \textsc{gen} & \textit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

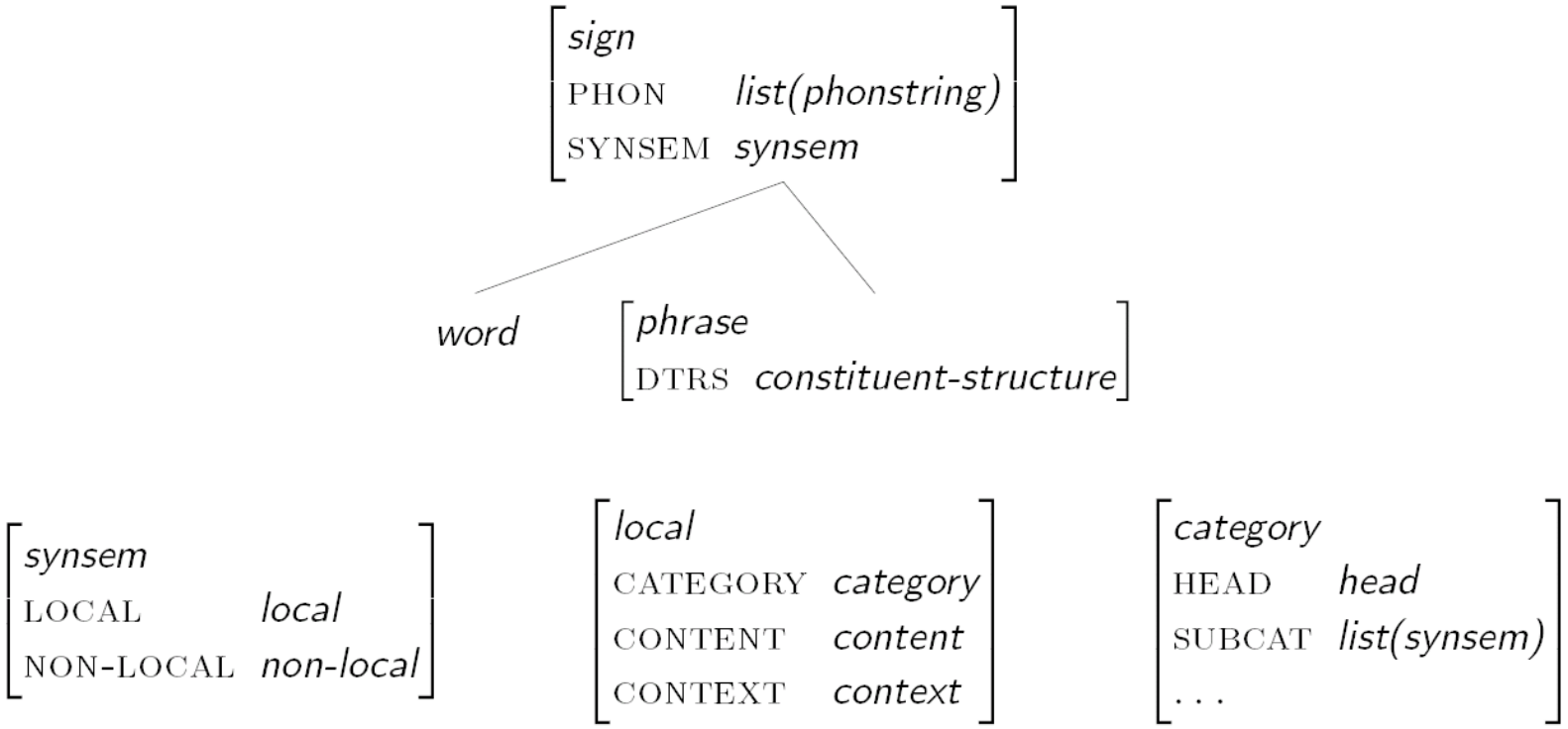
HPSG from a linguistic perspective (again)

Now that we have these feature structures, how do we use them for linguistic purposes?

- Specify a signature/ontology which allows us to make linguistically-relevant distinctions and puts appropriate features in the appropriate places

- Specify a theory which constrains that signature for a particular language
  - Lexicon specifies each word and the different properties that it has
    There can also be relations (so-called lexical rules) between words in the lexicon
  - Phrasal rules, or principles allow words to combine into phrases

An ontology of linguistic objects



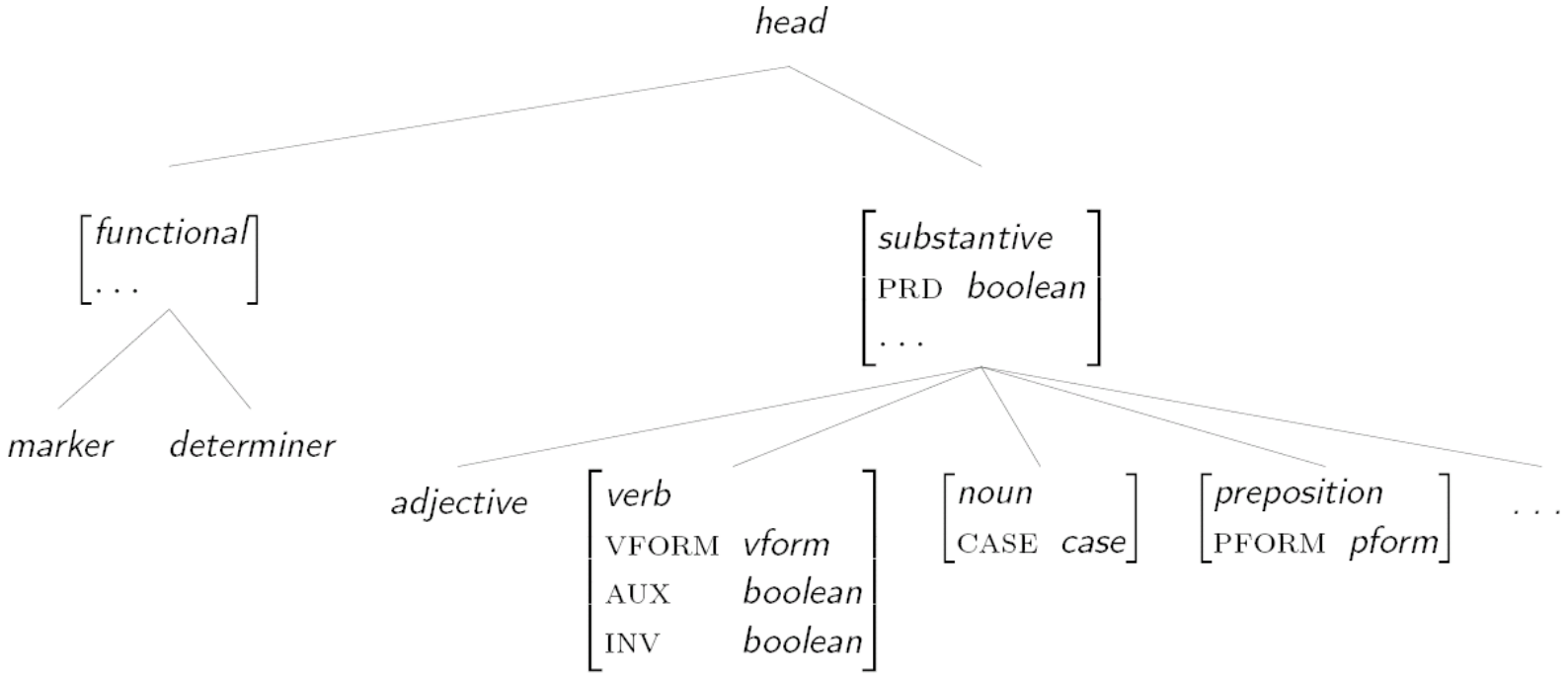
Why the complicated structure?

- LOCAL & NONLOCAL: Most linguistic constructions can be handled locally, but non-local constructions (e.g., extraction) require different mechanisms

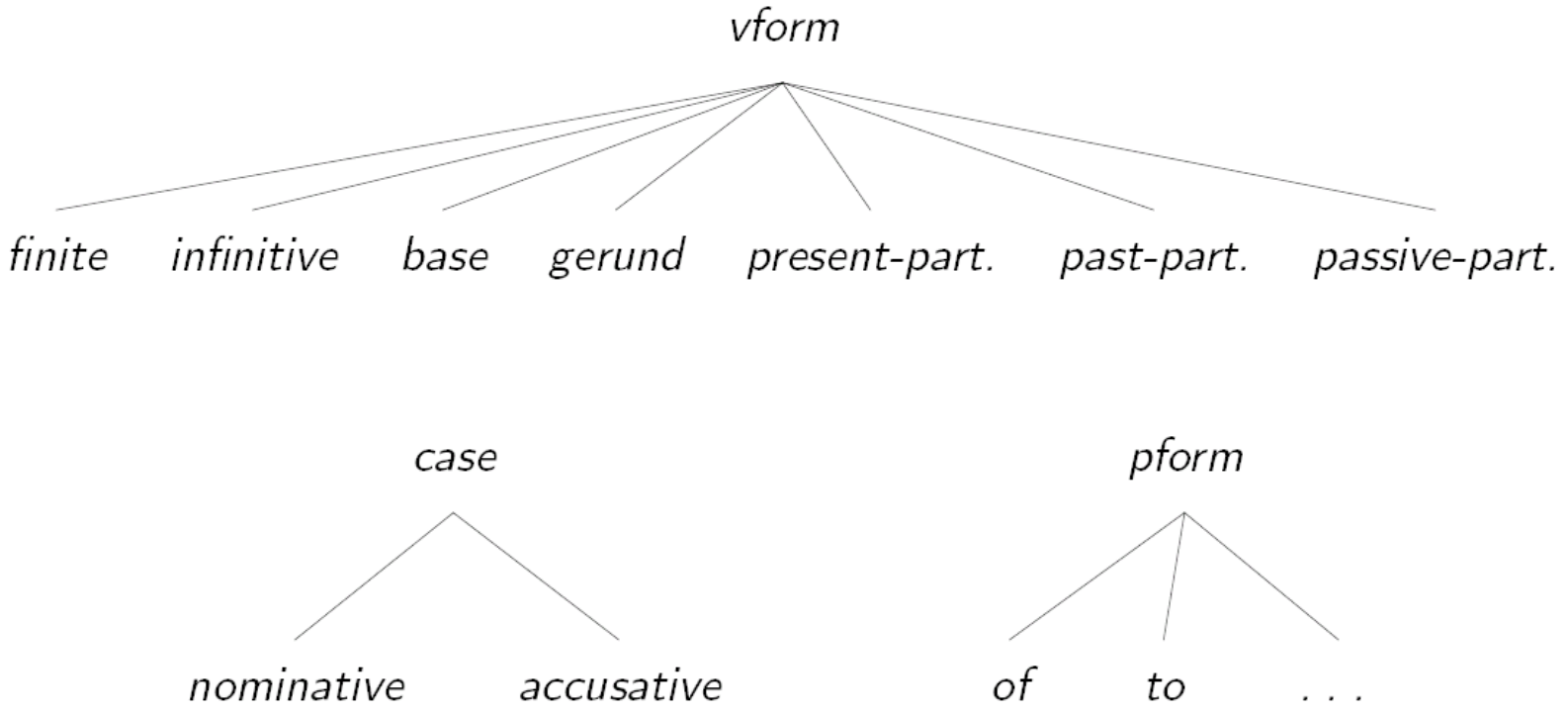
- CATEGORY, CONTENT, and CONTEXT: roughly, these correspond to syntactic, semantic, and pragmatic notions, all of which are locally determined

- HEAD and SUBCAT: a words syntactic information comes in two parts: its own lexical information (part of speech, etc.) and information about its arguments

Part-of-speech (head information)



Properties of particular part-of-speech



Motivating VFORM

(5) a. Peter will *win* the race. (base form)
b. * Peter will *won* the race.
c. * Peter will *to win* the race.

(6) a. Peter has *won* the race. (past participle)
b. * Peter has *win* the race.
c. Peter has *to win* the race.
(→ different verb)

(7) a. Peter seems *to win* the race. (*to*-infinitive)
b. * Peter seems *win* the race.
c. * Peter seems *won* the race.

Motivating CASE

(8) a. *He* left. (nom)
b. * *Him* left.

(9) a. She sees *him*. (acc)
b. * She sees *he*.

Motivating SUBCAT

(10) a. I *laugh*. (<NP>)
b. I *saw* him. (<NP NP>)
c. I *give* her the book. (<NP NP NP>)
d. I *said* that she left. (<NP S[that]>)

Cannot always be derived from semantics:

(11) a. Paul ate a steak. (<NP>)
b. Paul ate. (<NP NP>)
(12) a. Paul devoured a steak. (<NP>)
b. * Paul devoured (<NP NP>)

What SUBCAT does

The SUBCAT list can be thought of as akin to a word's valency requirements

- Items on the SUBCAT list are ordered by obliqueness—akin to LFG—not necessarily by linear order
- The SUBCAT Principle, described below, will describe a way for a word to combine with its arguments
  - That is, we will still need a way to go from the SUBCAT specification to some sort of tree structure

Locality of SUBCAT

SUBCAT selects a list of SYNSEM values, not SIGN values.

- If you work through the ontology, this means that a word does not have access to the DTRS list of items on its own SUBCAT list
- Intuitively, this means that a word cannot dictate properties of the daughters of its daughters.

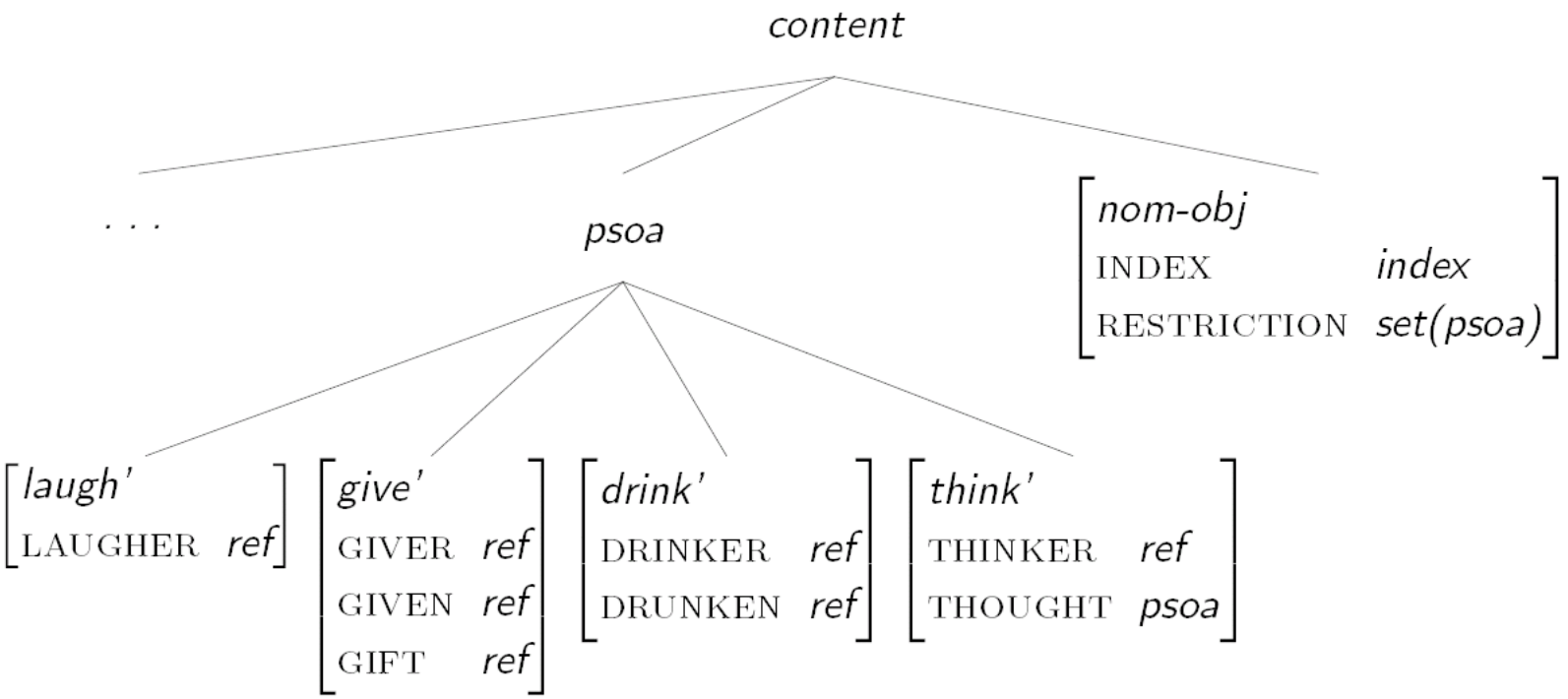
$\Rightarrow$ Constructions are thus restricted to local relations

CONTENT information

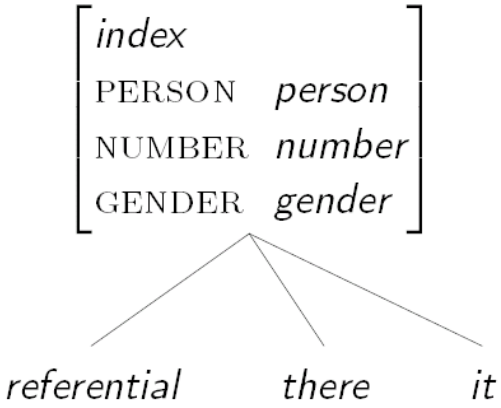
The CONTENT feature specifies different semantic information

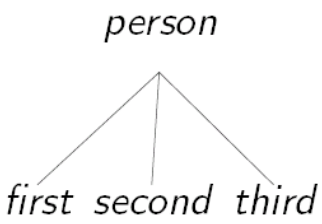
- A feature appropriate for *nominal-object* objects (a subtype of *content* objects) is INDEX
- Agreement features can be stated through the INDEX feature
- Note that CASE was put somewhere else (within HEAD), so CASE agreement is treated differently than person, number, and gender agreement (at least in English)

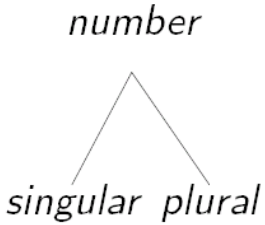
Semantic representations

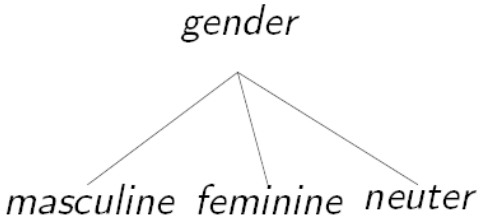


Indices


index
PERSON person
NUMBER number
GENDER gender
referential
there
it

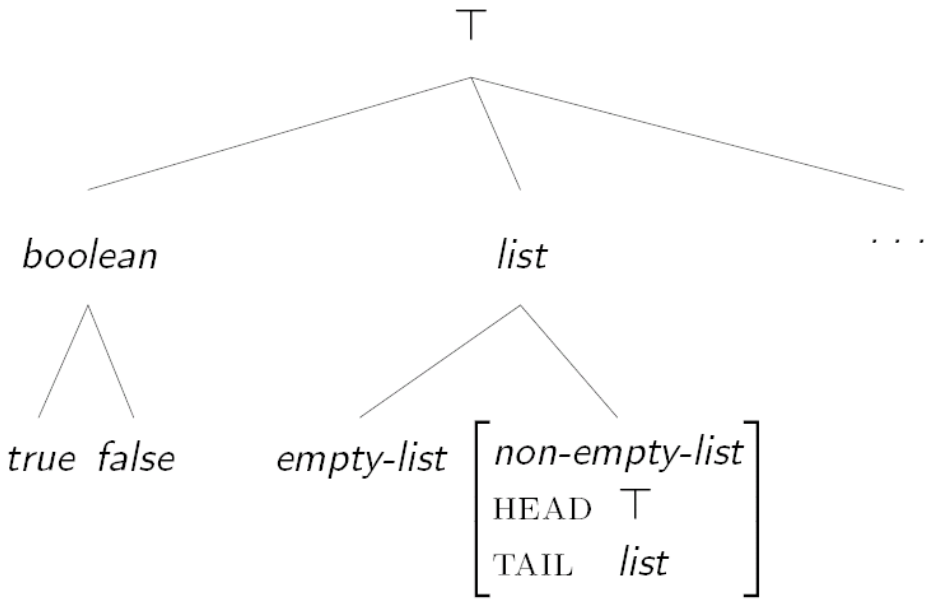

person
first
second
third


number
singular
plural


gender
masculine
feminine
neuter

Auxiliary data structures

Before we move on to some linguistic examples, a few other objects need to be defined



Abbreviations for describing lists

empty-list	is abbreviated as	*e-list*, <>
non-empty-list	is abbreviated as	*ne-list*
$\begin{bmatrix}\text{HEAD} & \boxed{1} \\ \text{TAIL} & \boxed{2}\end{bmatrix}$	is abbreviated as	$\langle \boxed{1} \mid \boxed{2} \rangle$
$\langle \ldots \boxed{1} \mid \langle\rangle \rangle$	is abbreviated as	$\langle \ldots \boxed{1} \rangle$
$\begin{bmatrix}\text{HEAD} & \boxed{1} \\ \text{TAIL} & \begin{bmatrix}\text{HEAD} & \boxed{2} \\ \text{TAIL} & \boxed{3}\end{bmatrix}\end{bmatrix}$	is abbreviated as	$\langle \boxed{1}, \boxed{2} \mid \boxed{3} \rangle$

Attention: $\langle \top \rangle$ and $\langle \boxed{1} \rangle$ describe all lists of length **one**!

Abbreviations of common AVMs

Pollard and Sag (1994) use some abbreviations to describe *synsem* objects:

Abbreviation	Abbreviated AVM
NP[1]	$\begin{bmatrix} \textit{synsem} \\ \text{LOCAL} \begin{bmatrix} \text{CATEGORY} \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONTENT} \mid \text{INDEX} \ \boxed{1} \end{bmatrix} \end{bmatrix}$
S:[1]	$\begin{bmatrix} \textit{synsem} \\ \text{LOCAL} \begin{bmatrix} \text{CATEGORY} \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONTENT} \ \boxed{1} \end{bmatrix} \end{bmatrix}$
VP:[1]	$\begin{bmatrix} \textit{synsem} \\ \text{LOCAL} \begin{bmatrix} \text{CATEGORY} \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \left\langle \textit{synsem} \right\rangle \end{bmatrix} \\ \text{CONTENT} \ \boxed{1} \end{bmatrix} \end{bmatrix}$

The Lexicon

The basic lexicon is defined by the *Word Principle* as part of the theory. It defines which of the ontologically possible words are grammatical:

$$\textit{word} \rightarrow \text{lexical-entry}_1 \vee \text{lexical-entry}_2 \vee \ldots$$

with each of the lexical entries being descriptions, such as e.g.:

$$\begin{bmatrix} \textit{word} & \\ \textsc{phon} & \langle \textit{laughs} \rangle \\ \textsc{synsem|loc} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{verb} \\ \textsc{vform} \;\; \textit{fin} \end{bmatrix} \\ \textsc{subcat} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{3rd,sing}] \right\rangle \end{bmatrix} \\ \textsc{content} & \begin{bmatrix} \textit{laugh}' \\ \textsc{laugher} \;\; \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

An example lexicon

$$
\textit{word} \rightarrow \left[\begin{array}{l}
\text{PHON } \langle \textit{gives} \rangle \\
\text{S|L} \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{l} \textit{verb} \\ \text{VFORM } \textit{fin} \end{array}\right] \\
\text{SUBCAT} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{sing}], \text{NP}[\textit{acc}]_{\boxed{2}}, \text{PP}[\textit{to}]_{\boxed{3}} \right\rangle
\end{array}\right] \\
\text{CONT} & \left[\begin{array}{l} \textit{give}' \\ \text{GIVER } \boxed{1} \\ \text{GIFT } \boxed{2} \\ \text{GIVEN } \boxed{3} \end{array}\right]
\end{array}\right]
\end{array}\right]
$$

$$\begin{bmatrix} \text{PHON} & \langle \textit{drinks} \rangle \\ \text{S|L} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{VFORM} \;\; \textit{fin} \end{bmatrix} \\ \text{SUBCAT} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{3rd,sing}],\ \text{NP}[\textit{acc}]_{\boxed{2}} \right\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \textit{drink'} \\ \text{DRINKER} \;\; \boxed{1} \\ \text{DRUNKEN} \;\; \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \langle \textit{drink} \rangle \\ \text{S|L} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{VFORM} \;\; \textit{fin} \end{bmatrix} \\ \text{SUBCAT} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{plur}],\ \text{NP}[\textit{acc}]_{\boxed{2}} \right\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \textit{drink'} \\ \text{DRINKER} \;\; \boxed{1} \\ \text{DRUNKEN} \;\; \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

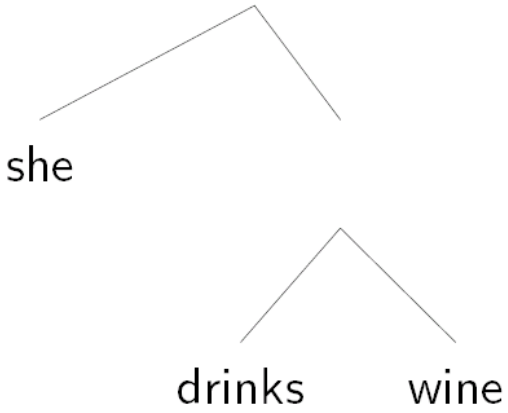
$$V\left[\begin{array}{ll}\text{PHON} & \langle \textit{she} \rangle \\ \text{SYNSEM|LOC} & \left[\begin{array}{ll}\text{CAT} & \left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{ll}\textit{noun} & \\ \text{CASE} & \textit{nom}\end{array}\right] \\ \text{SUBCAT} & \langle\rangle\end{array}\right] \\ \text{CONT} & \left[\text{INDEX} \left[\begin{array}{ll}\text{PER} & \textit{third} \\ \text{NUM} & \textit{sing}\end{array}\right]\right]\end{array}\right]\end{array}\right]$$

$$V\left[\begin{array}{ll}\text{PHON} & \langle \textit{wine} \rangle \\ \text{SYNSEM|LOC} & \left[\begin{array}{ll}\text{CAT} & \left[\begin{array}{ll}\text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle\end{array}\right] \\ \text{CONT} & \left[\text{INDEX} \left[\begin{array}{ll}\text{PER} & \textit{third} \\ \text{NUM} & \textit{sing}\end{array}\right]\right]\end{array}\right]\end{array}\right]$$

$$\mathrm{V}\left[\begin{array}{ll}\textsc{phon} & \langle to \rangle \\ \textsc{s|l} & \left[\begin{array}{ll}\textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \left[\begin{array}{l}preposition \\ \textsc{pform}\ \ to\end{array}\right] \\ \textsc{subcat} & \left\langle \mathrm{NP}[acc]_{\boxed{1}} \right\rangle\end{array}\right] \\ \textsc{cont} & \left[\textsc{index}\ \boxed{1}\right]\end{array}\right]\end{array}\right]$$

$$\mathrm{V}\left[\begin{array}{ll}\textsc{phon} & \langle think \rangle \\ \textsc{s|l} & \left[\begin{array}{ll}\textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \left[\begin{array}{l}verb \\ \textsc{vform}\ \ fin\end{array}\right] \\ \textsc{subcat} & \left\langle \mathrm{NP}[nom]_{\boxed{1}}[plur],\ \mathrm{S}[fin]{:}\boxed{2} \right\rangle\end{array}\right] \\ \textsc{cont} & \left[\begin{array}{l}think' \\ \textsc{thinker}\ \ \boxed{1} \\ \textsc{thought}\ \ \boxed{2}\end{array}\right]\end{array}\right]\end{array}\right]$$

A very first sketch of an example



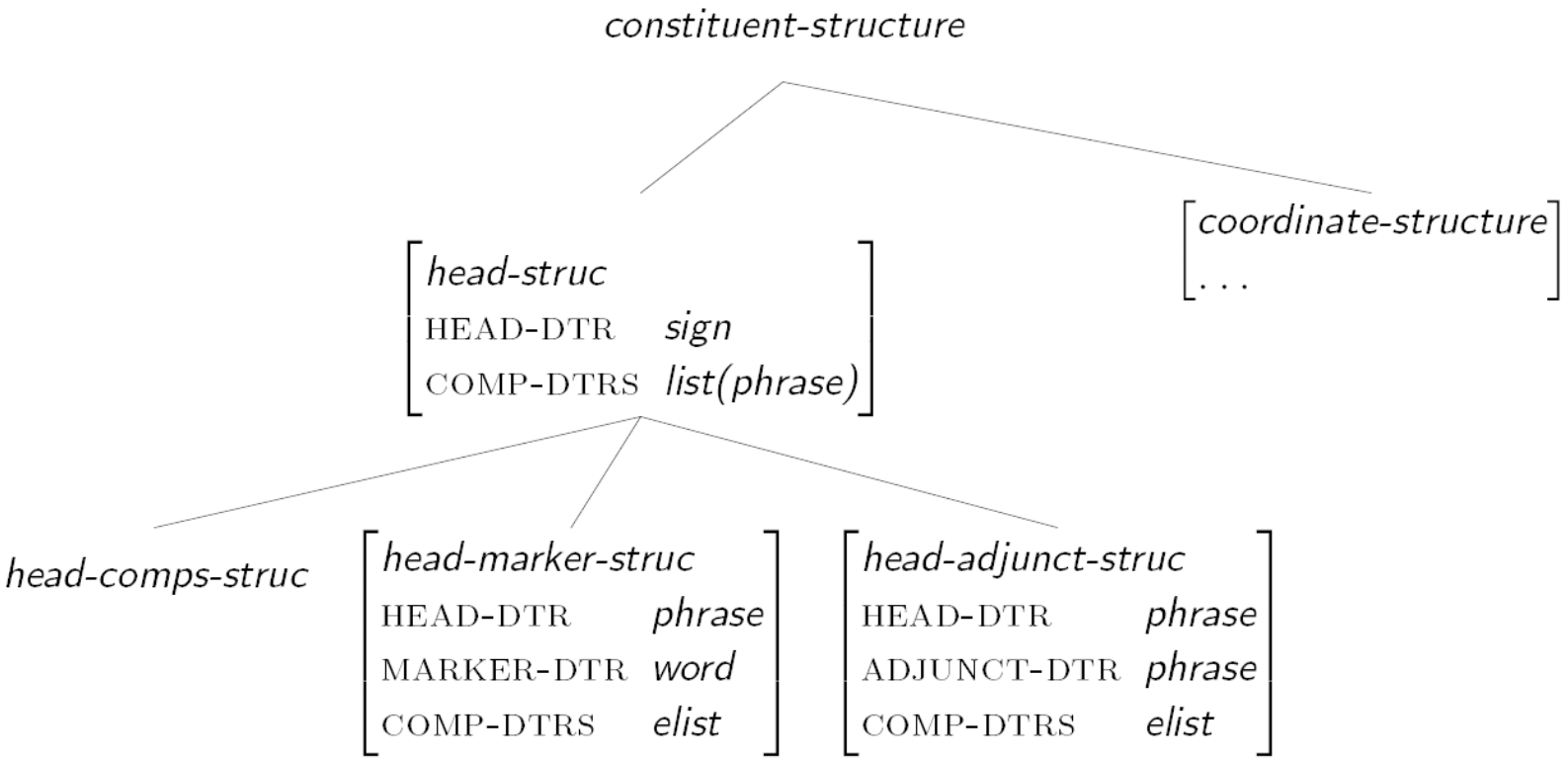
Types of phrases

In order to put words from our lexicon into a sentence, we have to define what makes an acceptable sentence structure

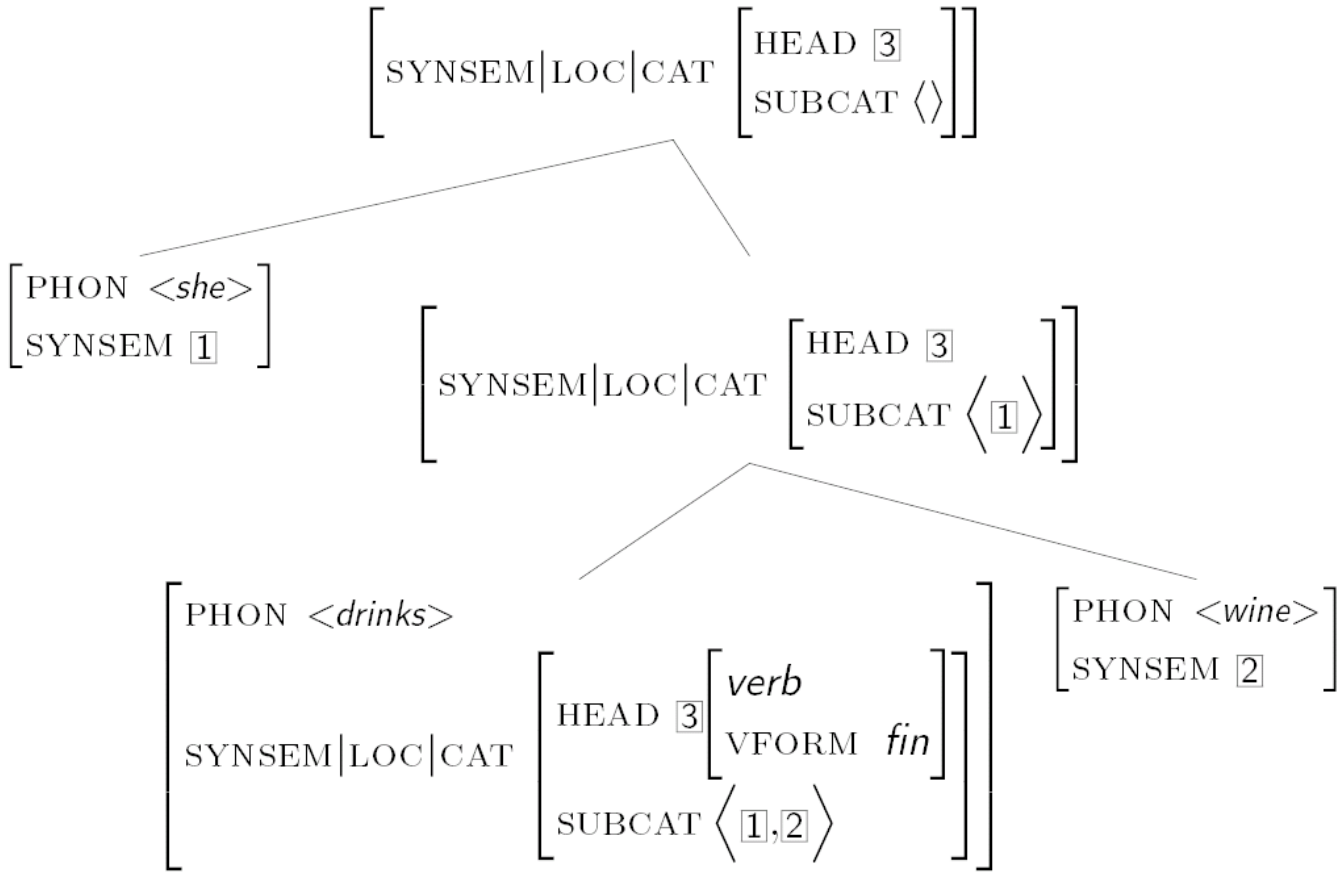
- Each *phrase* has a DTRS attribute (*words* do not have this attribute), which has a *constituent-structure* value
- This DTRS value loosely corresponds to what we normally view in a tree as daughters

  Additionally, tree branches" contain grammatical role information (adjunct, complement, etc.)
- By distinguishing different kinds of *constituent-structures*, we define what kinds of phrases exist in a language

An ontology of phrases



Sketch of an example for head-complement structures



Universal Principles

But how exactly did that last example work?

- *drinks* has head information specifying that it is a verb and so forth, and it also has subcategorization information specifying that it needs a subjects and an object.
  - The head information gets percolated up (The HEAD Principle)
  - The subcategorization information gets "checked off" as you move up in the tree (The SUBCAT Principle)

Such principles are treated as linguistic universals in HPSG.

Head-Feature Principle:

- In prose: The HEAD feature of any headed phrase is structure-shared with the HEAD value of the head daughter.

- Specified as a **constraint**:

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS} \;\; \textit{headed-structure} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD} & \boxed{1} \\ \text{DTRS}|\text{HEAD-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD} & \boxed{1} \end{bmatrix}$$

Subcat Principle:

In a headed phrase, the SUBCAT value of the head daughter is the concatenation of the phrase's SUBCAT list with the list (in order of increasing obliqueness of SYNSEM values of the complement daughters.

$$\left[\text{DTRS } \textit{headed-structure}\right] \rightarrow \left[\begin{array}{l}\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{SUBCAT } \boxed{1} \\ \text{DTRS } \left[\begin{array}{l}\text{HEAD-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{SUBCAT } \boxed{1} \oplus \boxed{2} \\ \text{COMP-DTRS } \textit{synsem2sign}\left(\boxed{2}\right)\end{array}\right]\end{array}\right]$$

with $\oplus$ standing for list concatenation, i.e., *append*, defined as follows

$$\begin{array}{lll} \textit{e-list} \oplus \boxed{1} & := & \boxed{1}. \\ \left[\begin{array}{ll}\text{FIRST} & \boxed{1} \\ \text{REST} & \boxed{2}\end{array}\right] \oplus \boxed{3} & := & \left[\begin{array}{lll}\text{FIRST} & \boxed{1} & \\ \text{REST} & \boxed{2} & \oplus \boxed{3}\end{array}\right]. \end{array}$$

Fallout from these Principles

- Note that agreement is handled neatly, simply by the fact that the SYNSEM values of a word's daughters are token-identical to the word's SUBCAT items.

One question remains before we can get the structure we have above:

- How exactly do we decide on a syntactic structure?
- i.e., Why is it that the object was checked off low and the subject was checked off at a higher point?

Answer: because of the ID schemata used

Immediate Dominance (ID) Schemata

- There is an inventory of valid ID schemata in a language
- Every headed phrase must satisfy exactly one of the ID schemata
  - Which ID schema is used depends on the type of the DTRS attribute
  - this goes back to the ontology of phrases we saw earlier

Immediate Dominance Principle (for English):

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS} \;\; \textit{headed-struc} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SYNSEM}|\text{LOC}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \left(\begin{bmatrix} \textit{verb} \\ \text{INV} \;\; - \end{bmatrix} \vee \neg \; \textit{verb} \right) \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \textit{head-comps-struc} \\ \text{HEAD-DTR} \;\; \textit{phrase} \\ \text{COMP-DTRS} \;\; \langle \textit{sign} \rangle \end{bmatrix} \end{bmatrix} \quad \text{(Head-Subject)}$$

$$\vee \begin{bmatrix} \text{SYNSEM}|\text{LOC}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \left(\begin{bmatrix} \textit{verb} \\ \text{INV} \;\; - \end{bmatrix} \vee \neg \; \textit{verb} \right) \\ \text{SUBCAT} & \langle \textit{synsem} \rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \textit{head-comps-struc} \\ \text{HEAD-DTR} \;\; \textit{word} \end{bmatrix} \end{bmatrix} \quad \text{(Head-Complement)}$$

$$\vee \begin{bmatrix} \text{SYNSEM}|\text{LOC}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{INV} \;\; + \end{bmatrix} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \textit{head-comps-struc} \\ \text{HEAD-DTR} \;\; \textit{word} \end{bmatrix} \end{bmatrix} \quad \text{(Head-Subject-Complement)}$$

$\vee$ . . . continued on next page

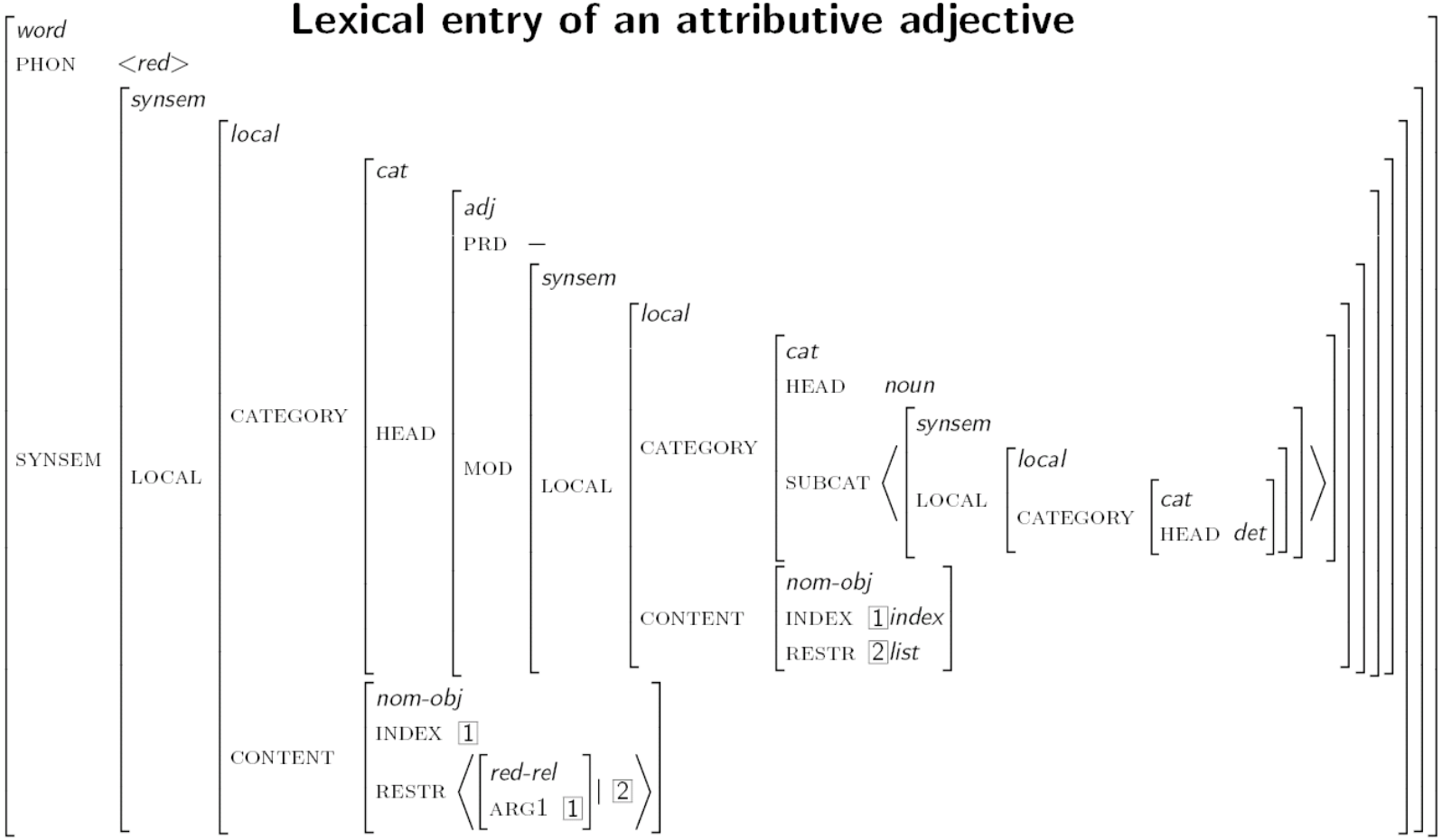
Immediate Dominance Principle (for English):

$$\begin{bmatrix} \textit{phrase} \\ \textsc{dtrs}\ \ \textit{headed-struc} \end{bmatrix} \rightarrow \begin{array}{l} \vdots \\ \vdots \\ \vdots \\ \vee \begin{bmatrix} \textsc{dtrs} & \begin{bmatrix} \textit{head-marker-struc} \\ \textsc{marker-dtr}|\textsc{synsem}|\textsc{loc}|\textsc{cat}|\textsc{head}\ \ \textit{marker} \end{bmatrix} \end{bmatrix} \text{(Head-Marker)} \\ \vee \begin{bmatrix} \textsc{dtrs} & \begin{bmatrix} \textit{head-adjunct-struc} \\ \textsc{adj-dtr}|\textsc{synsem}|\textsc{loc}|\textsc{cat}|\textsc{head}|\textsc{mod}\ \boxed{1} \\ \textsc{head-dtr}|\textsc{synsem}\ \ \boxed{1} \end{bmatrix} \end{bmatrix} \text{(Head-Adjunct)} \end{array}$$

So, in the example of *She drinks wine*, the DTRS value over *drinks wine* is a *head-comps-struc*, while the DTRS over the whole sentence is a *head-subj-struc*

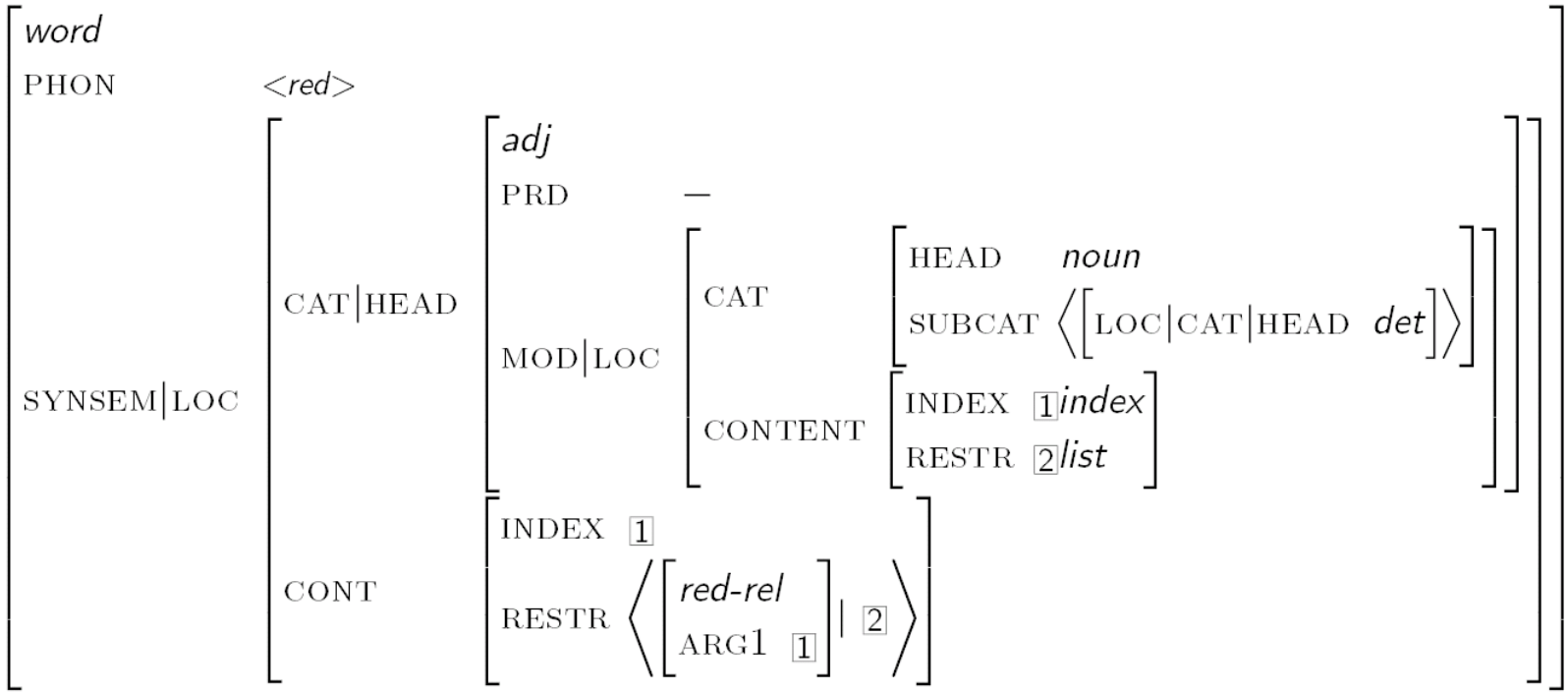
Towards Head Adjunct Structures

Lexical entry of an attributive adjective

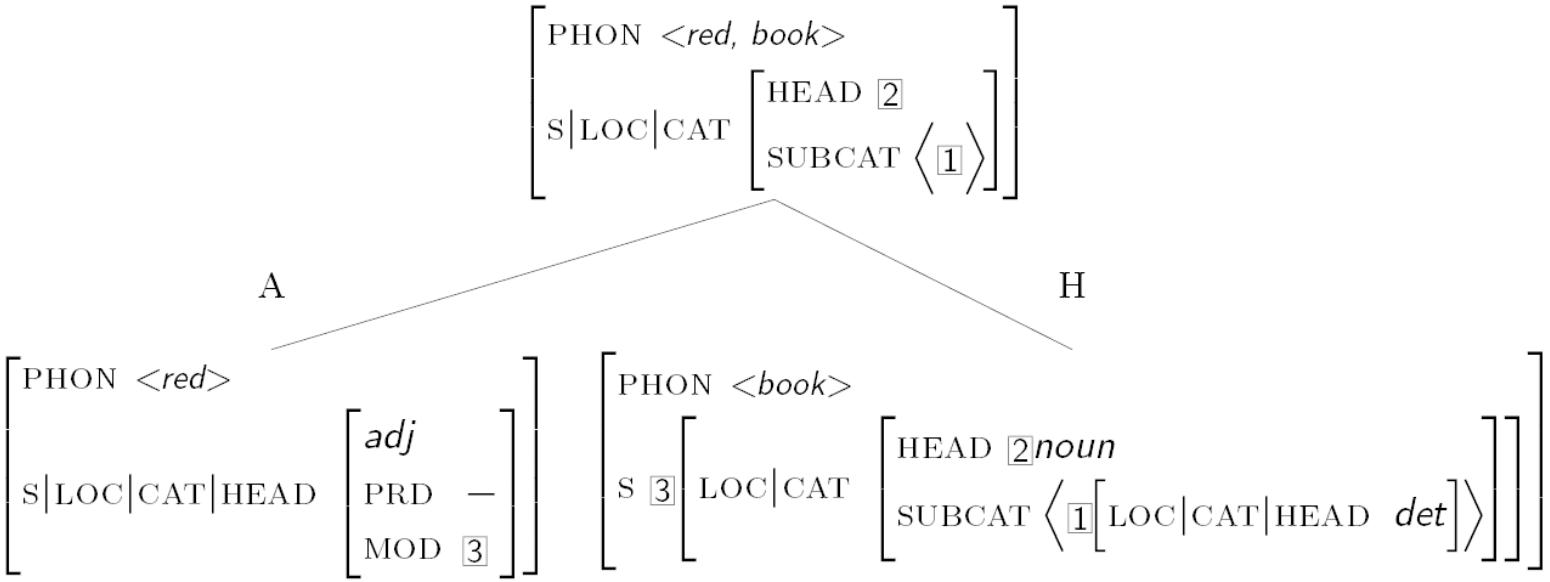


Lexical entry of an attributive adjective

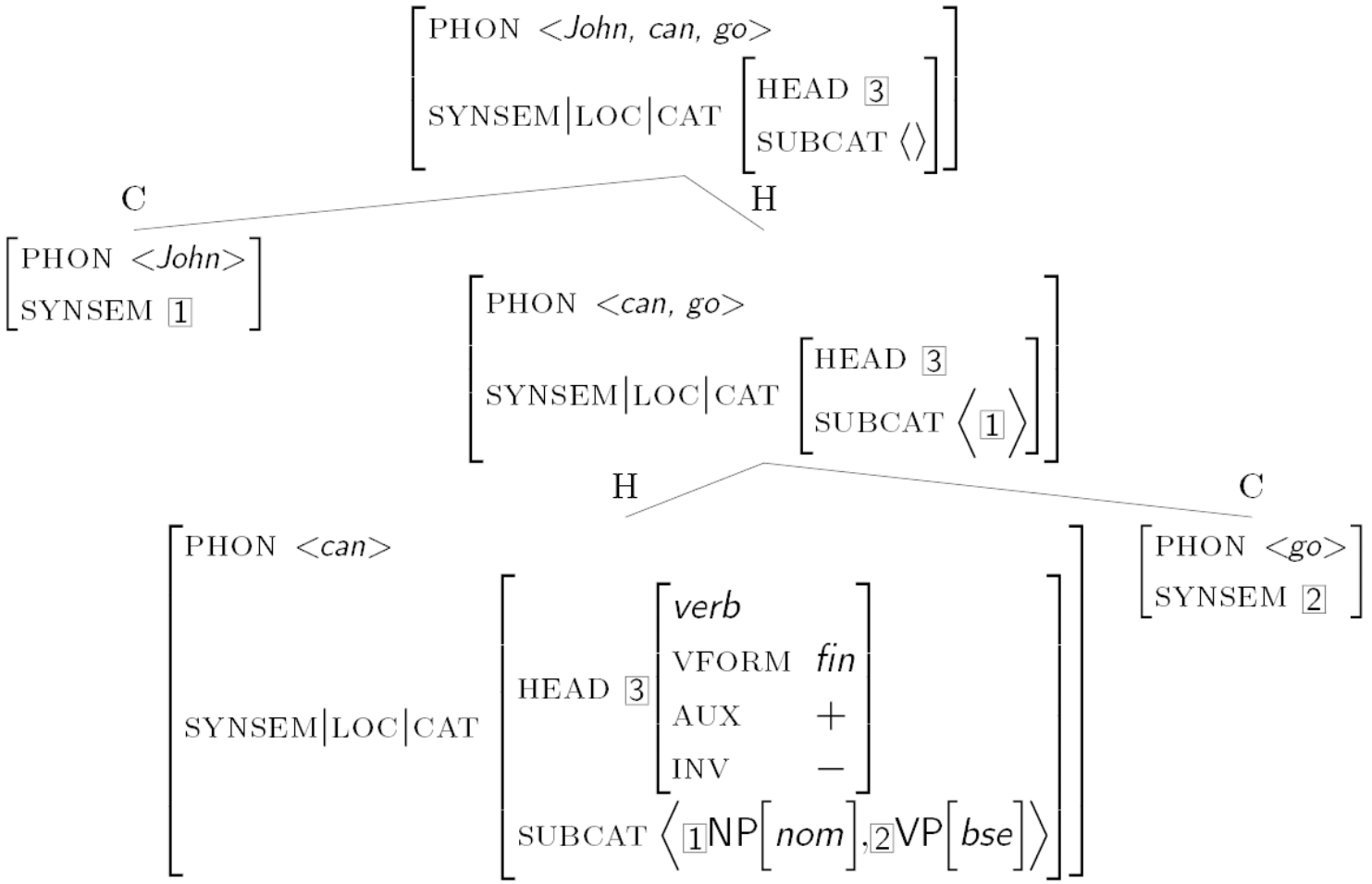
Version without redundant specifications



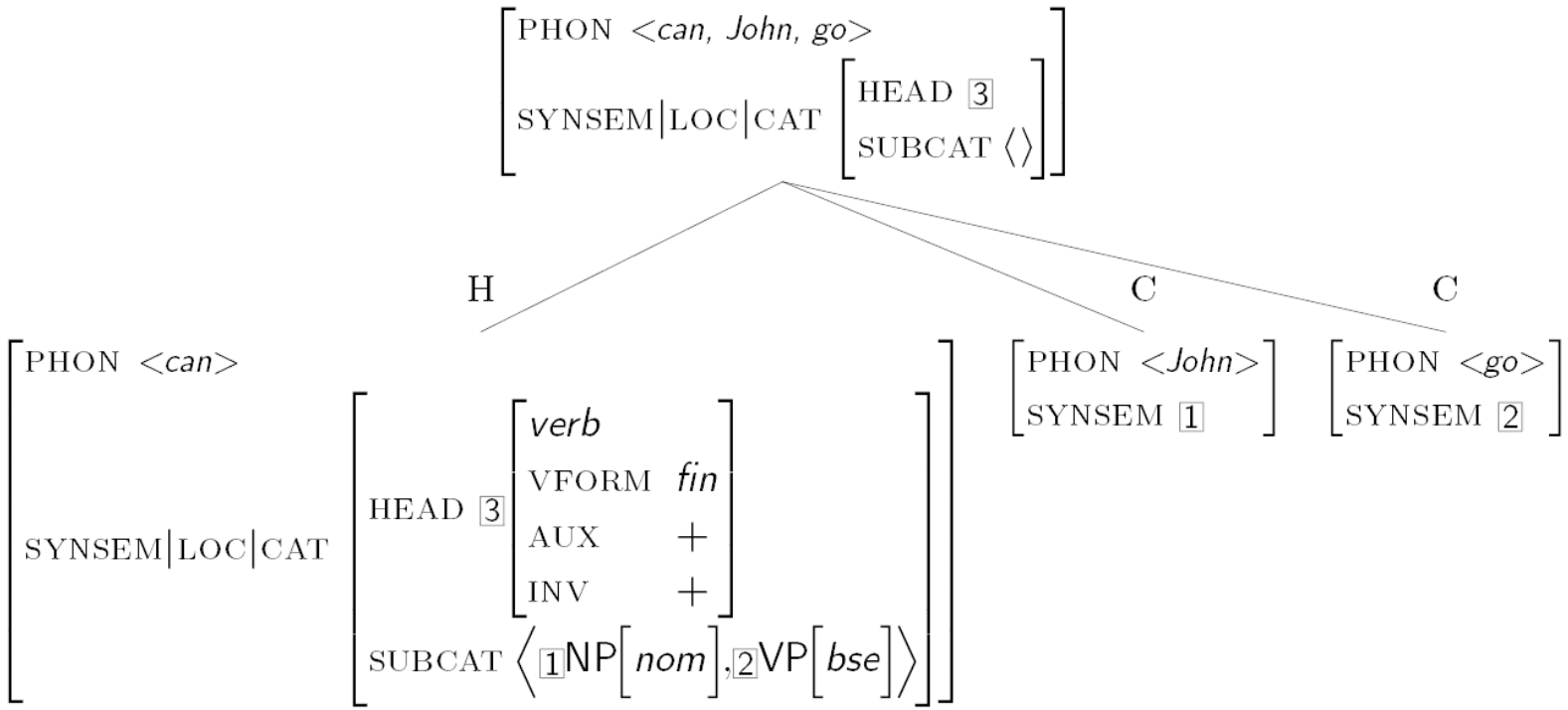
Sketch of an example for a head-adjunct structure



Sketch of an example with an auxiliary



Sketch of an example with an inverted auxiliary



SPEC Principle:

$$\left[\begin{array}{l} \textit{phrase} \\ \text{DTRS}\ \left[\left(\text{MARKER-DTR} \vee \text{COMP-DTRS}|\text{FIRST}\right)|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD}\ \textit{functional}\right] \end{array}\right]$$

$$\rightarrow \left[\text{DTRS}\ \left[\begin{array}{ll} \left(\text{MARKER-DTR} \vee \text{COMP-DTRS}|\text{FIRST}\right)|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD}|\text{SPEC} & \boxed{1} \\ \text{HEAD-DTR}|\text{SYNSEM} & \boxed{1} \end{array}\right]\right]$$

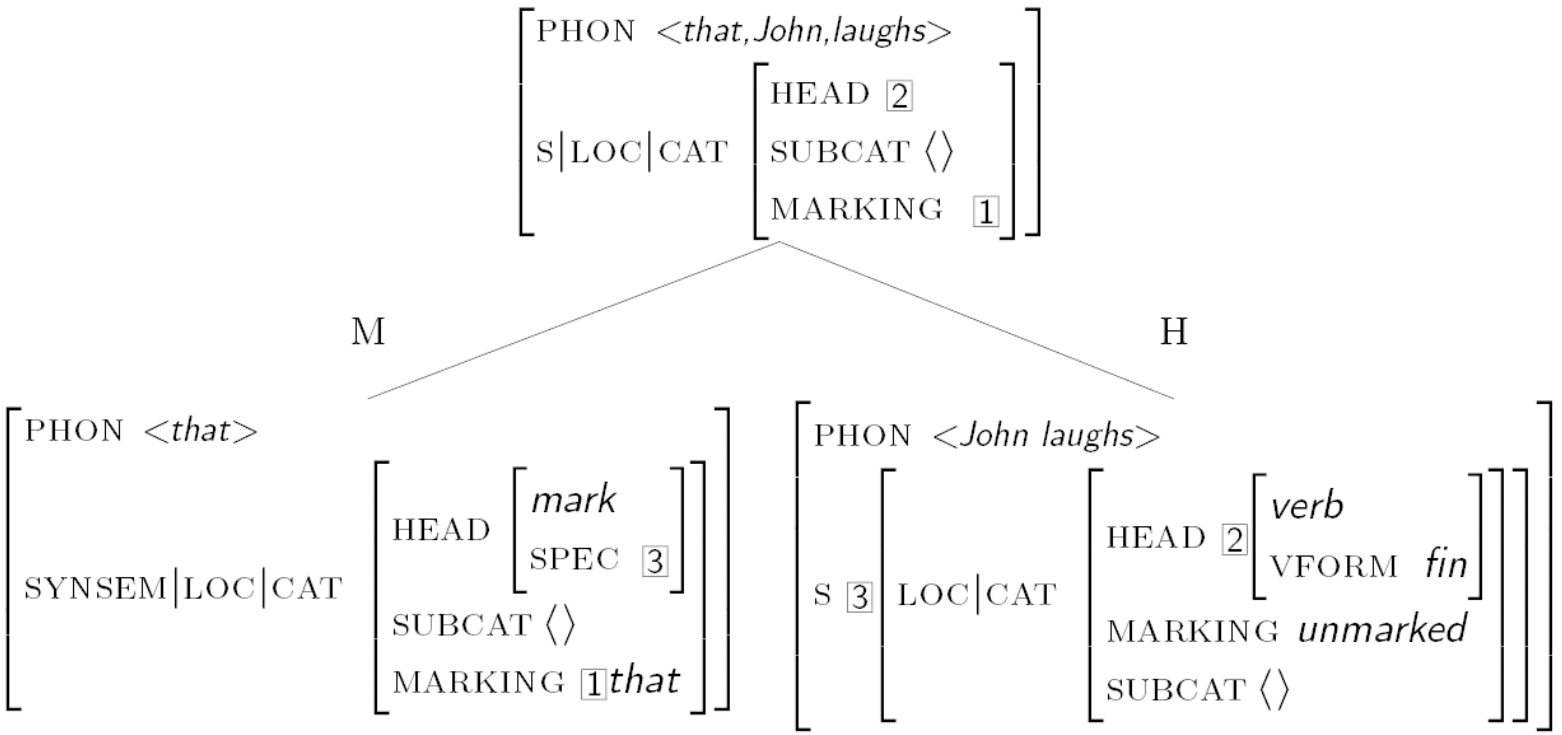
Marking Principle:

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS} \quad \textit{headed-structure} \end{bmatrix} \rightarrow \left[\begin{array}{l} \text{SYNSEM}|\text{LOC}|\text{CAT}|\text{MARKING}\ \boxed{1} \\ \text{DTRS}\ \begin{bmatrix} \textit{head-mark-struc} \\ \text{MARKER-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{MARKING}\ \boxed{1} \end{bmatrix} \end{array}\right] \vee \left[\begin{array}{l} \text{SYNSEM}|\text{LOC}|\text{CAT}|\text{MARKING}\ \boxed{1} \\ \text{DTRS}\ \begin{bmatrix} \neg\textit{head-mark-struc} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{MARKING}\ \boxed{1} \end{bmatrix} \end{array}\right]$$

Lexical entry of the marker *that*

$$\begin{bmatrix} \textsc{phon} & \langle \textit{that} \rangle \\ \textsc{synsem|loc|cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{mark} \\ \textsc{spec} & \begin{bmatrix} \textsc{loc|cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{verb} \\ \textsc{vform} & \textit{fin} \vee \textit{bse} \end{bmatrix} \\ \textsc{marking} & \textit{unmarked} \\ \textsc{subcat} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \textsc{subcat} & \langle\rangle \\ \textsc{marking} & \textit{that} \end{bmatrix} \end{bmatrix}$$

Sketch of an example for a head-marker structure



A few more points on HPSG

- We can view a grammar as a set of **constraints**: formulas which have to be true in order for a feature structure to be well-formed

  With such a view, parsing with HPSG falls into the realm of **constraint-based processing**

- Two important points about relating descriptions are subsumption and unification, loosely defined as:
  - **subsumption**: the description F subsumes the description G iff G entails F; i.e., F is more general than G
  - **unification**: the description of F and G unify iff their values are compatible
- **Closed World Assumption**: there are no linguistic species beyond what is specified in the type hierarchy