Head-Driven Phrase Structure Grammar (HPSG)

L614: Alternative Syntactic Theories
Spring 2010
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

Historical note: HPSG is based on Generalized Phrase Structure Grammar (GPSC) (Gazdar et al. 1985)
HPSG grammars from a linguistic perspective

From a linguistic perspective, an HPSG grammar consists of

a) a lexicon
   licensing basic words (which are themselves complex objects)

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 (typed) 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)
Skeleton of a typed feature structure

In attribute-value matrix (AVM) form, here is the skeleton of an object:

\[
\begin{align*}
\text{sign} & \quad \text{list(\text{PHON})} \\
\text{PHON} & \quad \left[ \begin{array}{c}
\text{synsem} \\
\text{SYNSEM} & \text{LOCAL} \quad \text{local} \\
\text{DTRS} & \text{list(\text{SIGN})}
\end{array} \right] \\
\text{NON-LOCAL} & \quad \text{non-local}
\end{align*}
\]
**Abbreviated skeleton**

Things are often abbreviated when written down (although, the object itself still contains the same things):

\[
\begin{array}{c}
\text{PHON} \ 	ext{list}\left(\text{PHON}\right) \\
\text{SYNSEM} \ 	ext{LOC} \quad \text{loc} \\
\text{DTRS} \ 	ext{list}\left(\text{SIGN}\right)
\end{array}
\]
An example tree

Let’s walk through an example to illustrate how feature structures can be used, starting with this rather impoverished tree:

```
she
  / \   /
drinks wine
```
Example tree with feature structures
Some things to note about the tree

- Phonology (PHON) is kept separate from syntax and semantics (SYNSEM), allowing different processes to operate on them.

- We say that *drinks* is a finite verb by specifying its type (verb) and that the value of its VFORM feature is *fin*.

- We have some way to say that parts of the tree share identical information, e.g., that a VP and its head daughter V have many of the same properties.

- We use lists to encode subcategorization information, and these items are identified with elements in the tree—note, too, how selection is kept local.
Even though it is called Head-driven Phrase Structure Grammar, the name is a misnomer

- Nothing about the formalism forces you to use PS trees

- In fact, technically, there are no trees as such, only features which encode objects akin to trees
  - Types license particular schemata (e.g., head-comps-struc), and a DTRS list keeps track of the constituent daughters
  - For ease of representation, we often display things in trees
  - But the example two slides back is more accurately represented as on the next slide
Lexicalized grammar

How do we start deriving such complex representations?

• One tenet of HPSG (akin to LFG) is that the lexicon contains complex representations of words

• So, when words are built into phrases, we have all this information at our hands

We can see this on the lexical entry on the next page, taken from Levine and Meurers (2005):

• For example, we can see that each word relates its syntactic argument structure (VALENCE) with its semantics (CONTENT)
Lexical entry for put

```
word
<put>

PHON

CATEGORY

VERB

VALENCE

HEAD

AUX

SUBJ

COMPS

put-relation

PUTTER

THING-PUT

DESTINATION

content

1

2

3

2

comps

D

NP

E

PP

E

3

3

3

3

3

3

3

3

3

3
```
Capturing dependencies

A grammatical framework needs to be able to capture the different grammatical dependencies of natural languages (cf. Levine and Meurers 2005, p. 5)

- Local dependencies: limited syntactic domain and largely lexical in nature
- Non-local dependencies: arbitrarily large syntactic domain and independent of lexicon

HPSG seems well-suited for this
Local dependencies

As with the other frameworks we’ve looked at, HPSG deals with local dependencies via the selectional properties of lexical heads (*head-driven*).

For example:

- Raising verbs select for an argument with which they share a subject

- Control (or equi) verbs select for an argument which has a co-indexed subject
Raising verb example
Control/Equi verb example

\[
\text{word} \quad \text{PHON} \\
\langle \text{try} \rangle \\
\text{SYNSEM} \mid \text{LOCAL} \\
\text{COMPS} \\
\text{try'} \\
\text{CONTENT} \\
\text{TRYER} \[1\] \\
\text{TRIED} \[2\] \\
\text{SUBJ} \langle \text{NP} \[1\] \rangle \\
\text{LOC} \\
\text{CAT} \mid \text{VAL} \\
\text{HEAD} \quad \text{verb} \\
\text{COMPS} \langle \rangle \\
\text{CONT} \[2\] \\
\text{SUBJ} \langle \text{NP} \[1\] \rangle \\
\text{COMPS} \langle \rangle \]
\]
Non-local dependencies

Instead of using transformations, HPSG analyzes unbounded dependency constructions (UDCs) by linking a filler with a gap

• Analysis relies on the feature SLASH

• The general idea is:
  – Trace lexical entry puts its local contents into a non-local SLASH set
  – This information is shared among the nodes in a tree
  – When the filler is realized, the information is removed from the SLASH set
HPSG grammars from a formal perspective

As with other frameworks we’ve examined, HPSG sets out to model the domain:

- Models of empirically observable objects need to be established, and
- Theories need to constrain which models actually exist.

Thus, 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:

```
object
  └── head
    ├── boolean
    └── subst(antive)
        ├── noun
        └── verb ...
    └── func(ctional)
        └── marker determiner
```
Sort-resolved

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

(1) a. \textit{func}  
   b. \textit{marker} \lor \textit{determiner}

That is, a type (or sort) is really a disjunction of its \textit{maximally specific subtypes}. 

22
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 attribute has an appropriate value.
    * Note that this is different from LFG.
  - 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)

\[
\begin{align*}
\text{word} & \quad \langle \text{walks} \rangle \\
\text{PHON} & \\
\text{SYNSEM|LOC} & \\
\text{CONTENT} & \\
\text{NP[nom][3rd,sing]} & \\
\text{walk'} & \\
\text{WALKER} & \\
\end{align*}
\]

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** descriptions 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., \([\text{SPOUSE} \ [\text{NAME} \ \text{mary}]\])

- **Tags** (structure sharing) to specify token identity, e.g. ✧
Descriptions (cont.)

Complex descriptions are obtained by combining descriptions with the help of conjunction ($\land$), disjunction ($\lor$) 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 \textit{spal.}
   I_{masc.sg} \textit{slept}_{masc.sg}

   b. On \textit{spal.}
   He_{masc.sg} \textit{slept}_{masc.sg}

(3) On the verb’s \textsc{subj} list:
   \[\begin{array}{c}
   \text{\textsc{word}} \\
   \text{\textsc{synsem|loc}} \\
   \text{\textsc{content}} \\
   \text{\textsc{index}} \\
   \text{\textsc{num sing gen masc}}
   \end{array}\]

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

Feature structure descriptions have **subsumption** relations between them.

- A more general description subsumes a more specific one.
- A more general description usually means that less features are specified.
Subsumption example

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

(4) a. 
\[
\begin{array}{c}
\text{word} \\
\text{SYNSEM|LOC} \\
\text{CAT|HEAD} \\
\text{CONTENT} \\
\text{INDEX} \\
\text{PER} \\
\text{NUM} \\
\text{GEN} \\
1st \\
\text{sing} \\
masc
\end{array}
\]

b. 
\[
\begin{array}{c}
\text{word} \\
\text{SYNSEM|LOC} \\
\text{CAT|HEAD} \\
\text{CONTENT} \\
\text{INDEX} \\
\text{PER} \\
\text{NUM} \\
\text{GEN} \\
3rd \\
\text{sing} \\
masc
\end{array}
\]
HPSG from a linguistic perspective (again)

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

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

2. 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
A tour of Pollard and Sag (1994)

We’ll start with the signature and theory from Pollard and Sag (1994).

In the next series of slides, you should:

- begin to understand what everything means

- begin to understand the connection between linguistic theory and its formalization in HPSG

- begin to gain an appreciation for a completely worked-out theory
An ontology of linguistic objects

\[ \text{sign} \]
\[ \text{PHON} \quad \text{list(phonstring)} \]
\[ \text{SYNSEM} \quad \text{synsem} \]

\[ \text{word} \]
\[ \text{phrase} \]
\[ \text{DTRS} \quad \text{constituent-structure} \]

\[ \text{synsem} \]
\[ \text{LOCAL} \quad \text{local} \]
\[ \text{NON-LOCAL} \quad \text{non-local} \]

\[ \text{local} \]
\[ \text{CATEGORY} \quad \text{category} \]
\[ \text{CONTENT} \quad \text{content} \]
\[ \text{CONTEXT} \quad \text{context} \]

\[ \text{category} \]
\[ \text{HEAD} \quad \text{head} \]
\[ \text{SUBCAT} \quad \text{list(synsem)} \]
\[ \ldots \]
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 word's 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

\[
\begin{array}{c}
vform \\
\downarrow \\
\text{finite} \quad \text{infinitive} \quad \text{base} \quad \text{gerund} \quad \text{present-part.} \quad \text{past-part.} \quad \text{passive-part.}
\end{array}
\]

\[
\begin{array}{c}
\text{case} \\
\downarrow \\
nominative \quad \text{accusative}
\end{array} \quad \quad \quad
\begin{array}{c}
pform \\
\downarrow \\
of \quad \text{to} \quad \ldots
\end{array}
\]
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.

NB: Here, we will use a single SUBCAT list, but later we will switch to a VALENCE feature, which contains both a SUBJ and COMPS list.
Locality of \textsc{subcat}

\textsc{subcat} selects a list of \textsc{synsem} values, not \textsc{sign} values.

- If you work through the ontology, this means that a word does not have access to the \textsc{dtrs} list of items on its own \textsc{subcat} list.

- Intuitively, this means that a word cannot dictate properties of the daughters of its daughters.

⇒ Constructions are thus restricted to local relations.
The **CONTENT** feature specifies different semantic information

- A feature appropriate for *nominal-object* objects (a subtype of *content* objects) is **INDEX** (as shown on the next slide)

- 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

content

psoa

nom-obj
INDEX index
RESTRICTION set(psoa)

[laugh']
[LAUGHER ref
[give']
[GIVER ref
[GIVEN ref
[GIFT ref
[drink']
[DRINKER ref
[DRUNKEN ref
[think']
[THINKER ref
[THOUGHT psoa]
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*

\[
\begin{array}{c}
\text{HEAD} \ 1 \\
\text{T A I L} \ 2
\end{array}
\]

is abbreviated as \(\langle 1 \mid 2 \rangle\)

\(\langle \ldots 1 \mid \langle \rangle \rangle\)

is abbreviated as \(\langle \ldots 1 \rangle\)

\[
\begin{array}{c}
\text{HEAD} \ 1 \\
\text{T A I L} \ \begin{array}{c}
\text{HEAD} \ 2 \\
\text{T A I L} \ 3
\end{array}
\end{array}
\]

is abbreviated as \(\langle 1, 2 \mid 3 \rangle\)

Attention: \(\langle T \rangle\) and \(\langle 1 \rangle\) describe all lists of length **one**!
# Abbreviations of common AVMs

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

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Abbreviated AVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td><code>synsem</code></td>
</tr>
<tr>
<td>S:</td>
<td><code>synsem</code></td>
</tr>
<tr>
<td>VP:</td>
<td><code>synsem</code></td>
</tr>
</tbody>
</table>

### NP

- **LOCAL**: CATEGORY [HEAD `noun` SUBCAT `⟨⟩`]
- **CONTENT|INDEX**: `⟨⟩`

### S:

- **LOCAL**: CATEGORY [HEAD `verb` SUBCAT `⟨⟩`]
- **CONTENT**: `⟨⟩`

### VP:

- **LOCAL**: CATEGORY [HEAD `verb` SUBCAT `⟨synsem⟩`]
- **CONTENT**: `⟨⟩`
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:

\[
\text{word} \rightarrow \text{lexical-entry}_1 \lor \text{lexical-entry}_2 \lor \ldots
\]

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

\[
\begin{align*}
\text{word} & \quad \text{PHON} \\
\text{SYNSEM} & \quad \text{LOC} \\
\text{CAT} & \quad \text{HEAD} \\
\text{SUBCAT} & \quad \text{CONTENT} \\
\text{NP} & \quad \text{nom} \\
\text{verb} & \quad \text{VFORM} \\
\text{laugh} & \quad \text{U} \\
\end{align*}
\]
An example lexicon

\[
\begin{align*}
\text{word} & \rightarrow \left[ \begin{array}{c}
\text{PHON} <\text{gives}> \\
\text{CAT} \\
\text{HEAD} \\
\text{VFORM} \ fin \\
\text{NP}[\text{nom}_1[\text{sing}], \ NP[\text{acc}_2], \ PP[\text{to}_3]]
\end{array} \right]
\end{align*}
\]
A very first sketch of an example

Here's that impoverished tree again:

We're going to see how the theory licenses this structure ...
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 \texttt{DTRS} attribute (*words* do not have this attribute), which has a *constituent-structure* value.

- This \texttt{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

constituent-structure

head-struc
HEAD-DTR sign
COMP-DTRS list(phrase)

head-comps-struc

head-marker-struc
HEAD-DTR phrase
MARKER-DTR word
COMP-DTRS enlist

head-adjunct-struc
HEAD-DTR phrase
ADJUNCT-DTR phrase
COMP-DTRS enlist

coordinate-structure

...
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**:

\[
\begin{align*}
\text{phrase} \\
\text{DTRS} \text{ headed-structure}
\end{align*} \rightarrow \begin{align*}
\text{SYNSEM|LOC|CAT|HEAD} \\
\text{DTRS|HEAD-DTR|SYNSEM|LOC|CAT|HEAD}
\end{align*}
\]
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).

\[
\begin{align*}
[DTRS \text{ headed-structure}] & \rightarrow \begin{bmatrix}
\text{SYNSEM}\mid\text{LOC}\mid\text{CAT}\mid\text{SUBCAT} [1] \\
\text{DTRS} \left[\begin{array}{c}
\text{HEAD-DTR} \\
\text{COMP-DTRS} \text{ synsem2sign}(2)
\end{array}\right]
\end{bmatrix}
\end{align*}
\]

with \(\oplus\) standing for list concatenation, i.e., \textit{append}, defined as follows

\[
\begin{align*}
e-list \quad & \oplus [1] := [1]. \\
[\text{FIRST } 1] \oplus [3] & := [\text{FIRST } 1] \oplus [3].
\end{align*}
\]
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 $d_{TRS}$ attribute
  - this goes back to the ontology of phrases we saw earlier

Formally, though, these constraints are phrased as the universal principles were
Immediate Dominance Principle (for English):

\[
\begin{align*}
\text{phrase} & \quad \text{DTRS} \quad \text{headed-struc} \\
\rightarrow & \\
\text{SYNSEM} & \mid \text{LOC} \mid \text{CAT} \\
\text{HEAD} & \quad \left( \begin{array}{c}
\text{verb} \\
\text{INV} \\
- \\
\end{array} \right) \quad \lor \quad \neg \text{verb} \\
\text{SUBCAT} & \quad \langle \rangle \\
\text{head-comps-struc} & \\
\text{HEAD-DTR} & \quad \text{phrase} \\
\text{COMP-DTR} & \quad \langle \text{sign} \rangle \\
\text{DTRS} & \\
\end{align*}
\]

(Head-Subject)

\[
\begin{align*}
\lor & \\
\text{SYNSEM} & \mid \text{LOC} \mid \text{CAT} \\
\text{HEAD} & \quad \left( \begin{array}{c}
\text{verb} \\
\text{INV} \\
- \\
\end{array} \right) \quad \lor \quad \neg \text{verb} \\
\text{SUBCAT} & \quad \langle \text{synsem} \rangle \\
\text{head-comps-struc} & \\
\text{HEAD-DTR} & \quad \text{word} \\
\text{DTRS} & \\
\end{align*}
\]

(Head-Complement)

\[
\begin{align*}
\lor & \\
\text{SYNSEM} & \mid \text{LOC} \mid \text{CAT} \\
\text{HEAD} & \quad \left( \begin{array}{c}
\text{verb} \\
\text{INV} \\
+ \\
\end{array} \right) \\
\text{SUBCAT} & \quad \langle \rangle \\
\text{head-comps-struc} & \\
\text{HEAD-DTR} & \quad \text{word} \\
\text{DTRS} & \\
\end{align*}
\]

(Head-Subject-Complement)

\[
\lor \ldots \text{continued on next page}
\]
Immediate Dominance Principle (for English):

\[
\begin{align*}
\text{phrase} \\
\text{DTRS headed-struc}
\end{align*}
\]

\[
\begin{align*}
\rightarrow & \\
\vdots & \\
\vdots & \\
\checkmark \\
\text{DTRS} \\
\text{head-marker-struc} \\
\text{MARKER-DTR|SYNSEM|LOC|CAT|HEAD marker}
\end{align*}
\]

\[
\begin{align*}
\checkmark \\
\text{DTRS} \\
\text{head-adjunct-struc} \\
\text{ADJ-DTR|SYNSEM|LOC|CAT|HEAD|MOD II} \\
\text{HEAD-DTR|SYNSEM II}
\end{align*}
\]

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:

\[
\begin{align*}
\text{phrase} & \quad \left[ \text{DTRS} \left[ \left( \text{MARKER-DTR} \lor \text{COMP-DTRS}\mid \text{FIRST}\right)\left| \text{SYNSEM}\mid \text{LOC}\mid \text{CAT}\mid \text{HEAD } \text{functional} \right. \right] \right] \\
\rightarrow & \quad \left[ \text{DTRS} \left[ \left( \text{MARKER-DTR} \lor \text{COMP-DTRS}\mid \text{FIRST}\right)\left| \text{SYNSEM}\mid \text{LOC}\mid \text{CAT}\mid \text{HEAD}\mid \text{SPEC} \right. \right] \right] \\
& \quad \left[ \text{HEAD-DTR}\mid \text{SYNSEM} \right]
\end{align*}
\]
Marking Principle:

\[
\begin{align*}
\text{phrase} & \quad \xrightarrow{\text{DTRS}} \quad \text{headed-structure} \\
\quad & \quad \rightarrow \\
\quad & \quad \begin{cases}
\text{SYNSEM|LOC|CAT|MARKING} \\
\text{DTRS} \quad \text{head-mark-struc} \\
\text{SYNSEM|LOC|CAT|MARKING} \\
\quad \lor \\
\quad \begin{cases}
\text{DTRS} \quad \neg \text{head-mark-struc} \\
\text{HEAD-DTR|SYNSEM|LOC|CAT|MARKING}
\end{cases}
\end{cases}
\end{align*}
\]
Lexical entry of the marker *that*
Sketch of an example for a head-marker structure

\[
\begin{align*}
\text{PHON} & \quad \langle \text{that, John, laughs} \rangle \\
\text{S|LOC|CAT} & \quad \text{HEAD} \quad 2 \\
& \quad \text{SUBCAT} \quad \langle \rangle \\
& \quad \text{MARKING} \quad 1 \\
\end{align*}
\]

\[
\begin{align*}
\text{PHON} & \quad \langle \text{that} \rangle \\
\text{SYNSEM|LOC|CAT} & \quad \text{HEAD} \\
& \quad \text{SPEC} \quad 3 \\
& \quad \text{SUBCAT} \quad \langle \rangle \\
& \quad \text{MARKING} \quad 1 \text{that} \\
\end{align*}
\]

\[
\begin{align*}
\text{PHON} & \quad \langle \text{John laughs} \rangle \\
\text{S} \quad 3 \\
\text{LOC|CAT} & \quad \text{HEAD} \quad 2 \\
& \quad \text{VFORM fin} \\
& \quad \text{MARKING unmarked} \\
& \quad \text{SUBCAT} \quad \langle \rangle \\
\end{align*}
\]
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