Syntactic Formalisms

Pascal Amsili (slides from Markus Disckinson)

Sorbonne Nouvelle, Lattice (CNRS/ENS-PSL)

Cogmaster, october 2020

Slides borrowed to Markus Dickinson

(`md7` AT `edu.indiana` (flipped around, of course))

Class: "Alternative Syntactic Theories" (L614 Spring 2010)

Overview

Syntactic analysis

- **Generative grammar** = collection of words and rules with which we generate strings of those words, i.e., sentences
- Syntax attempts to capture the nature of those rules

(1) Colorless green ideas sleep furiously.
(2) *Furiously sleep ideas green colorless.

- What generalizations are needed to capture the difference between grammatical sentences and ungrammatical sentences?

Syntax: What does it mean?

Can view a syntactic theory in a number of ways, two of which are the following:

- Psychological model: syntactic structures correspond to what is in the heads of speakers
- Computational model: syntactic structures are formal objects which can be mathematically manipulated.

$\Rightarrow$ We will focus on the computational way of viewing grammar for this class

Formalism vs. theory

Will we actually look at theories? ... Sort of.

- A **theory** describes a set of data and makes predictions for new data
  - In this class, we will emphasize theories which are **testable**, i.e., can be verified or falsified
- A **formalism** provides a way of defining a theory with mathematical rigor
  - It is essentially a set of beliefs and conditions that frame how generalizations can be made.

The course name (*Alternative Syntactic Theories*) is a bit of a misnomer: we will actually be focusing on *formalisms*, and we will use theories to exemplify them.

The Transformational tradition

Roughly speaking, **transformational syntax** (GB, P&P, ...) has focused on the following:

- Explanatory adequacy: the data must fit with a deeper model, that of universal grammar
- Psychological: does the grammar make sense in light of what we know of how the mind works?
- Universality: generalizations must be applicable to all languages
- Transformations/Movement: (surface) sentences are derived from underlying other sentences, e.g. passives are derived from active sentences

But this kind of theory often doesn't lend itself well to computational applications

Alternative assumptions

- Prioritize descriptive adequacy over explanatory adequacy
- Prioritize computational effectiveness over psychological reality
  - e.g., movement is disfavored
- Prioritize description in one language before dealing with all languages

The data will always be the same, but how you handle it, as we'll see, depends largely upon your assumptions

Overview

Making it computational

How is a grammatical theory useful for computational lingusitics?

- Parsing: take an input sentence and return the syntactic analysis and/or state whether it is a valid sentence
- Generation: take a meaning representation and generate a valid sentence

$\Rightarrow$ Both tasks are often subparts of practical applications (e.g., dialogue systems)

Computational needs

To use a grammar for parsing or generation, we need to have a grammar that meets several criteria:

- Accurate: gives a correct analysis
- Precise: tells a computer exactly what it is that you want it to do
- Efficient: able to parse a sentence and return one or only a small number of parses
- Useful: is relatively easy to map a syntactic structure of a sentence to its meaning

⇒ These needs are not necessarily why the computational formalisms were developed, but they are some of the reasons why people use them.

Computational Grammar Formalisms

The formalisms we will look at this quarter generally share several properties:

- Descriptive adequacy
- Precise encodings (implementable)
- Constrained mathematical formalisms
- Monostratal frameworks
- (Usually) highly lexical

Descriptive adequacy

Some researchers try to explain the underlying mechanisms, but we are most concerned with being able to *describe* linguistic phenomena

- Provide a structural description for every well-formed sentence
  - Define which sentences are well-formed and which are not in a language
- Give us an accurate encoding of a language
- Interested in broad-coverage, i.e., can (try to) describe all of a language

  → less of a distinction between core and periphery phenomena

Precise encodings

Mathematical formalism: formal way to generate sets of strings

Precisely define:

- elementary structures
- ways of combining those structures

$\Rightarrow$ Such an emphasis on mathematical precision makes these grammar formalisms more easily implementable

- Will 2 parts of your grammar conflict?
- If we have precisely encoded the grammar, we can answer this question with certainty.

11/25

Constrained mathematical formalisms

Formalism should (arguably) be **constrained**, i.e., cannot be allowed to specify all strings

- Linguistic motivation: Limits the scope of the theory of grammar
- Computational motivation: Allows us to define efficient processing models

This is different than constraining a theory

- What is the minimum amount of mathematical overhead that we need to describe language?

Monostratal frameworks

Only have one (surface) syntactic level

- Make no recourse to movement or transformations
- Augment your basic (phrase structure) tree with information that can describe "movement" phenomena
  - Need some way to relate different structures (e.g., active and passive) without invoking, e.g., traces

$\Rightarrow$ Without having to refer to movement, easier to process sentences on a computer.

Lexical

In the past, rules applied to broad classes and only some information was put in the lexicon, e.g., subcategorization information.

But more and more theories emphasize the role of individual lexical items in grammatical constructions

- Linguistic motivation: lexicon best way to specify some generalizations: *He told/*divulged me the truth*
- Computational motivation: can derive lexical information from corpora

⇒ Shift more of the information to the lexicon; each lexical item may be a complex object.

Brief mention of complexity

We have touched on the complexity of different formalisms

Type	Automaton		Grammar	
	Memory	Name	Rule	Name
0	Unbounded	TM	$\alpha \rightarrow \beta$	General rewrite
1	Bounded	LBA	$\beta\ A\ \gamma \rightarrow \beta\ \delta\ \gamma$	Context-sensitive
2	Stack	PDA	$A \rightarrow \beta$	Context-free
3	None	FSA	$A \rightarrow xB,\ A \rightarrow x$	Right linear

- TM: Turing Machine
- LBA: Linear-Bounded Automaton
- PDA: Push-Down Automaton
- FSA: Finite-State Automaton

Criteria under which to evaluate grammar formalisms

There are three kinds of criteria:

- linguistic naturalness
- mathematical power
- computational effectiveness and efficiency

The weaker the type of grammar:

- the stronger the claim made about possible languages
- the greater the potential efficiency of the parsing procedure

Reasons for choosing a stronger grammar class:

- to capture the empirical reality of actual languages
- to provide for elegant analyses capturing more generalizations ($\rightarrow$ more "compact" grammars)

Overview

Are CFGs good enough?

- Data from Swiss German and other languages show that CFGs are not powerful enough to handle all natural language constructions
- CFGs are not easily lexicalized (and we need lexical knowledge)
- CFGs become complicated once we start taking into account agreement features, verb subcategorizations, unbounded dependency constructions, raising constructions, etc.

We need more refined formalisms ...

Beyond CFGs

We want to move beyond CFGs to better capture language, but maintain that level of precision

We can look at it a couple of ways:

- Extend the basic model of CFGS with, e.g., complex categories, functional structure, feature structures, ...
- Eliminate CFG model (or derive it some other way)

The frameworks we will investigate take one of these approaches

Various formalisms

What we plan to have a look at:

- Dependency Grammar (DG)
- Tree-Adjoining Grammar (TAG)
- Lexical-Functional Grammar (LFG)
- Head-driven Phrase Structure Grammar (HPSG)
- Combinatory Categorial Grammar (CCG)

Dependency Grammar (DG)

- The way to analyze a sentence is by looking at the relations between words
- No grouping, or constituency, is used
  - DG traditions are often completely independent of constituency-based traditions (e.g., CFGs)
  - DG is not a unified framework; there are a host of different frameworks within this tradition
- A verb and its arguments drive an analysis, which is closely related to the semantics of a sentence

Some of the other frameworks we'll investigate utilize insights from DG

Tree-Adjoining Grammar (TAG)

The analysis looks like a CFG tree, but the way to get it is completely different ...

- Elementary structures are trees of arbitrary height
- Trees are rooted in lexical items, i.e. lexicalized
  - In other words, the lexicon contains tree fragments as parts of lexical entries
- Put trees together by substituting and adjoining them, resulting in a final tree which looks like a CFG-derived tree

Lexical-Functional Grammar (LFG)

- Functional structure (subject, object, etc.) divided from constituent structure (tree structure)
  - Kind of like combining dependency structure with phrase structure
  - The f-structures are potentially very complex, however.
- Can express some generalizations in f-structure; some in c-structure;
  - i.e., not restricted to saying everything in terms of trees

Head-driven Phrase Structure Grammar (HPSG)

- Sentences, phrases, and words all uniformly treated as linguistic signs, i.e., complex objects of features
  - Many analyses rely on a CFG backbone, but this need not be so.
- Similar to LFG in its use of a feature architecture
- Uses an inheritance hierarchy to relate different types of objects (e.g., nouns and determiners are both types of nominals)

Combinatory Categorial Grammar (CCG)

- Categorial Grammar derives sentences in a proof-solving manner, maintaining a close link with a semantic representation
- Lexical categories specify how to combine words into sentences
  - The idea of selection is crucial, e.g., a verb will select for the number and type of arguments
  - Again, lexical entries contain tree-like information
- CCG has sophisticated mechnisms that deal nicely with coordination, extraction, and other constructions

Overview

Dependency Grammar

- Not a coherent grammatical framework: wide range of different kinds of DG
  - just as there are wide ranges of "generative syntax"
- Different core ideas than phrase structure grammar
- We will base a lot of our discussion on [Mel'čuk(1988)]

Dependency grammar is important for those interested in CL:

- Increasing interest in dependency-based approaches to syntactic parsing in recent years (e.g., CoNLL-X shared task, 2006)

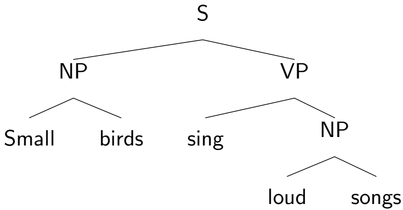
Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the (translated) words of Lucien Tesnière [Tesnière(1959)]:
  - The sentence is an *organized whole*, the constituent elements of which are *words*. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives *connections*, the totality of which forms the structure of the sentence. [1.3] The structural connections establish *dependency* relations between the words. Each connection in principle unites a *superior* term and an *inferior* term. [2.1] The superior term receives the name *governor*. The inferior term receives the name *subordinate*. Thus, in the sentence *Alfred parle* [. . .], *parle* is the governor and *Alfred* the subordinate. [2.2]

Overview: constituency

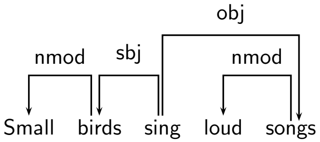
(1) Small birds sing loud songs

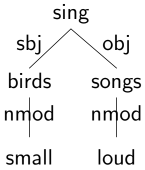
What you might be more used to seeing:



Overview: dependency

The corresponding dependency tree representations [Hudson(2000)]:





Constituency vs. Relations

- DG is based on relationships between words, i.e., **dependency relations**
  - A → B means *A governs B* or *B depends on A* ...
  - Dependency relations can refer to syntactic properties, semantic properties, or a combination of the two
    - → Some variants of DG separate syntactic and semantic relations by representing different layers of dependency structures
  - These relations are generally things like subject, object/complement, (pre-/post-)adjunct, etc.
    - Subject/Agent: *John* fished.
    - Object/Patient: Mary hit *John*.
- PSG is based on groupings, or constituents
  - Grammatical relations are not usually seen as primitives, but as being derived from structure

Simple relation example

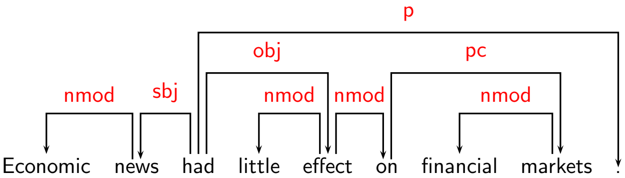
For the sentence *John loves Mary*, we have the relations:

- loves $\rightarrow_{\text{subj}}$ John
- loves $\rightarrow_{\text{obj}}$ Mary

Both *John* and *Mary* depend on *loves*, which makes *loves* the head, or **root**, of the sentence (i.e., there is no word that governs *loves*)

- The structure of a sentence, then, consists of the set of pairwise relations among words.

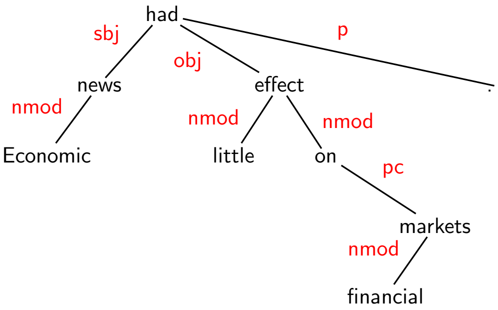
Dependency Structure



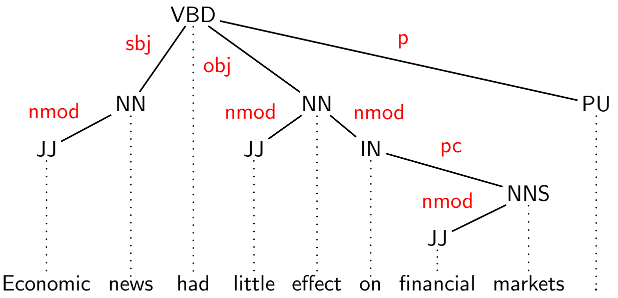
Terminology

Superior	Inferior
Head	Dependent
Governor	Modifier
Regent	Subordinate
⋮	⋮

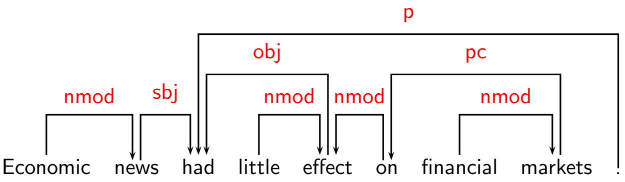
Notational Variants



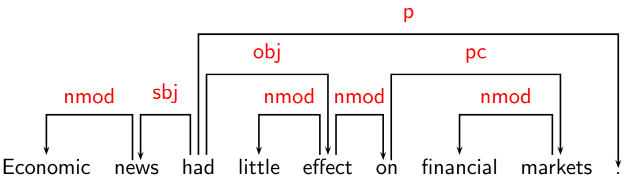
Notational Variants



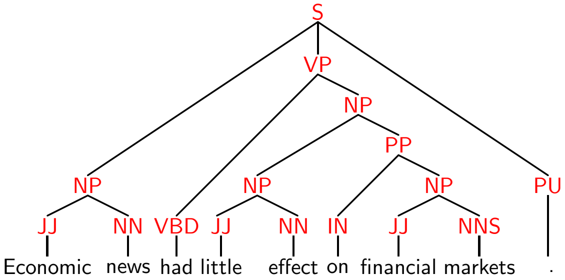
Notational Variants



Notational Variants



Phrase Structure



Comparison

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs),
  - functional categories (arc labels),
  - possibly some structural categories (parts-of-speech).
- Phrase structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions).
- Hybrid representations may combine all elements.

Some Theoretical Frameworks

- Word Grammar (WG) [Hudson(1984), Hudson(1990)]
- Functional Generative Description (FGD) [Sgall et al.(1986)Sgall, Hajičová and Panevová]
- Dependency Unification Grammar (DUG) [Hellwig(1986), Hellwig(2003)]
- Meaning-Text Theory (MTT) [Mel'čuk(1988)]
- (Weighted) Constraint Dependency Grammar ([W]CDG) [Maruyama(1990), Harper and Helzerman(1995), Menzel and Schröder(1998), Schröder(2002)]
- Functional Dependency Grammar (FDG) [Tapanainen and Järvinen(1997), Järvinen and Tapanainen(1998)]
- Topological/Extensible Dependency Grammar ([T/X]DG) [Duchier and Debusmann(2001), Debusmann et al.(2004)Debusmann, Duchier and Kruijff]

Overview

Some Theoretical Issues

- Dependency structure sufficient as well as necessary?
- Mono-stratal or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - Word forms?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - Grammatical functions?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?

Some Theoretical Issues

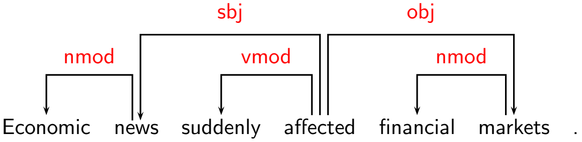
- Dependency structure sufficient as well as necessary?
- Mono-stratal or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - Word forms?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - Grammatical functions?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?

Criteria for Heads and Dependents

- Criteria for a syntactic relation between a head *H* and a dependent *D* in a construction *C* [Zwicky(1985), Hudson(1990)]:
  1. *H* determines the syntactic category of *C*; *H* can replace *C*.
  2. *H* determines the semantic category of *C*; *D* specifies *H*.
  3. *H* is obligatory; *D* may be optional.
  4. *H* selects *D* and determines whether *D* is obligatory.
  5. The form of *D* depends on *H* (agreement or government).
  6. The linear position of *D* is specified with reference to *H*.
- Issues:
  - Syntactic (and morphological) versus semantic criteria
  - Exocentric versus endocentric constructions

Some Clear Cases

Construction	Head	Dependent
Exocentric	Verb	Subject (sbj)
	Verb	Object (obj)
Endocentric	Verb	Adverbial (vmod)
	Noun	Attribute (nmod)



Some Tricky Cases

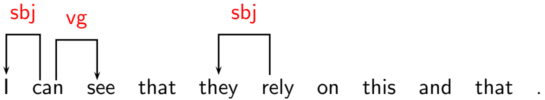
- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation

?

I can see that they rely on this and that .

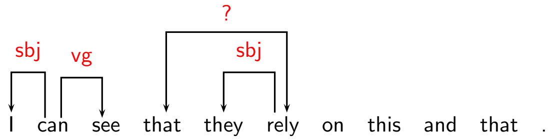
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



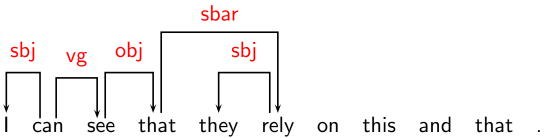
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



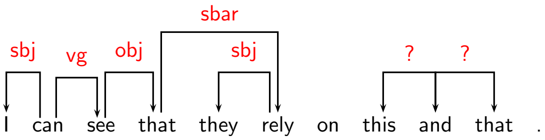
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



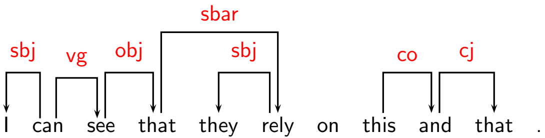
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



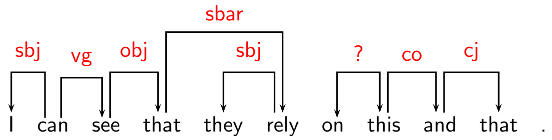
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



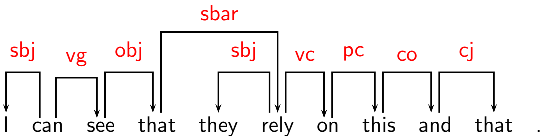
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



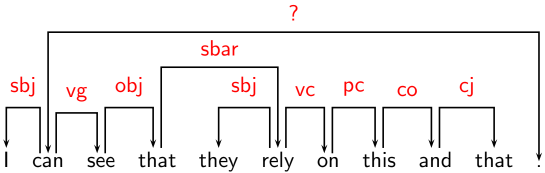
Some Tricky Cases

- Complex verb groups (auxiliary $\leftrightarrow$ main verb)
- Subordinate clauses (complementizer $\leftrightarrow$ verb)
- Coordination (coordinator $\leftrightarrow$ conjuncts)
- Prepositional phrases (preposition $\leftrightarrow$ nominal)
- Punctuation



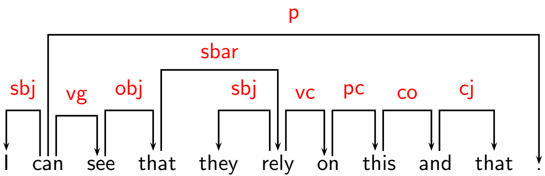
Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation



Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation

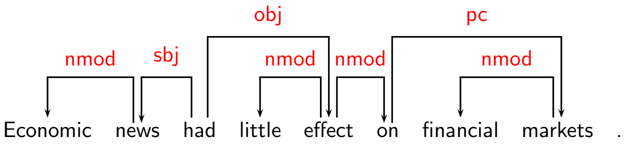


Dependency Graphs

- A dependency structure can be defined as a directed graph G, consisting of
  - a set V of nodes,
  - a set E of arcs (edges),
  - a linear precedence order $<$ on V (not in every theory)
- Labeled graphs:
  - Nodes in V are labeled with word forms (and annotation).
  - Arcs in E are labeled with dependency types.
- Notational conventions ($i, j \in V$):
  - $i \to j \;\equiv\; (i, j) \in E$
  - $i \to^* j \;\equiv\; i = j \vee \exists k : i \to k,\ k \to^* j$

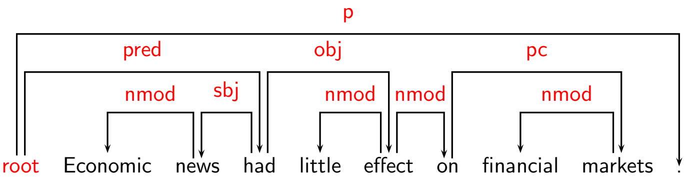
Formal Conditions on Dependency Graphs

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



Formal Conditions on Dependency Graphs

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.



Formal Conditions on Dependency Graphs

- G is (weakly) connected:
  - For every node i there is a node j such that $i \to j$ or $j \to i$.
- G is acyclic:
  - If $i \to j$ then not $j \to^* i$.
- G obeys the single-head constraint:
  - If $i \to j$, then not $k \to j$, for any $k \neq i$.
- G is projective:
  - If $i \to j$ then $i \to^* k$, for any k such that $i < k < j$ or $j < k < i$.

Projectivity

Projectivity (or, less commonly, **adjacency** [Hudson(1990)])

- A head (A) and a dependent (B) must be adjacent: A is adjacent to B provided that every word between A and B is a subordinate of A.

(2) with great difficulty

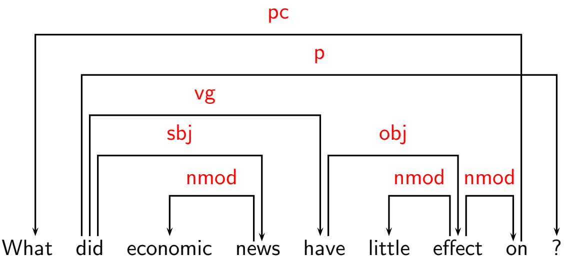
(3) *great with difficulty

- with $\rightarrow$ difficulty
- difficulty $\rightarrow$ great

**great with difficulty* is ruled out because branches would have to cross in that case

Projectivity

- Most theoretical frameworks do not assume projectivity.
- Non-projective structures are needed to account for
  - long-distance dependencies,
  - free word order.



Valency and Grammaticality

An important concept in many variants of DG is that of **valency** = the ability of a word to take arguments

A lexicon might look like the following [Hajič et al.(2003)Hajič, Panevová, Urešová, Bémová, Kolářová and Pajas]:

	Slot_1	Slot_2	Slot_3
sink$_1$	ACT(nom)	PAT(acc)	
sink$_2$	PAT(nom)		
give	ACT(nom)	PAT(acc)	ADDR(dat)

To determine grammaticality (roughly) ...

1. Words have valency requirements that must be satisfied
2. Apply general rules to the valencies to see if a sentence is valid

Capturing Adjuncts and Complements

There are two main kinds of dependencies for A $\rightarrow$ B:

- Head-Complement: if A (the head) has a slot for B, then B is a complement
- Head-Adjunct: if B has a slot for A (the head), then B is an adjunct

B is dependent on A in either case, but the selector is different

- The adjunct/complement distinction is captured in the type of dependency relation and/or in the lexicon

Layers of dependencies

[Mel'čuk(1988)] allows for different dependency layers

It looks like a subject depends on the verb, but the form of the verb depends on the subject (mutual dependence):

(4) a. The child is playing.
    b. The children are playing.

Solution:

- Dependence of *child*/*children* on the verb is syntactic
- Dependence of the verb(form) on the subject is morphological

Double dependencies

Likewise, here it seems that *clean* depends both on the verb *wash* and on the noun *dish*

(5) Wash the dish *clean*.

Solution:

- Dependence of *clean* on *wash* is syntactic (cf. case)
- Dependence of *clean* on *dish* is semantic (cf. gender)

(6) My našli zal pust-ym
We found the hall$_{masc}$ empty$_{masc.sg.inst}$

Double dependencies (2)

Hudson's Word Grammar [Hudson(2004)] explicitly allows for **structure-sharing**, explicitly violating the single-head constraint:

- wash $\rightarrow$ clean
- dish $\rightarrow$ clean

NB: Hudson also uses this to account for non-projectivity

Overview

1. Introduction
2. Dependency Grammars
   - Presentation
   - Discussion
   - Conclusion
3. Tree Adjoing Grammar
4. Lexical-functional Grammar
5. Head-Driven PS Grammar

Sorbonne Nouvelle

Relation to phrase structure

What is the relation between DG and PSG?

- If a PS tree has heads marked, then you can derive the dependencies
- Likewise, a DG tree can be converted into a PS tree by grouping a word with its dependents
  - But what the constituents are is still open (binary-branching, flat)
  - And phrases are not categorized

Advantages and Disadvantages of DG

Advantages:

- Close connection to semantic representation
- More flexible structure for, e.g., non-constituent coordination
- Easier to capture some typological regularities
- Vast & expanding body of computational work on dependency parsing

Disadvantages:

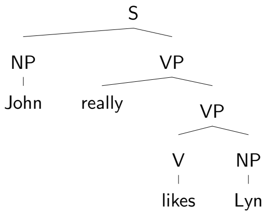
- No constituents makes analyzing coordination difficult
- No distinction between modifying a constituent vs. an individual word
- Harder to capture things like, e.g., subject-object asymmetries

Overview

TAG

- Pseudo-extension of CFGs
  - Abandon the context-free grammar formalism
  - Keep the idea of deriving complete trees in a sequence of rewriting steps—but in TAG we rewrite *trees*, not strings
- Highly lexicalized (LTAG):
  - Every tree is associated with exactly one lexical item
  - Every lexical item is associate with a set of trees

Phrase Structure Trees



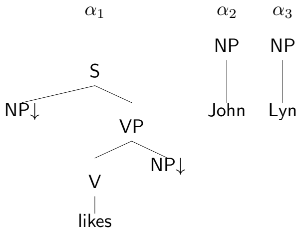
(1) a. S → NP VP
b. VP → really VP
c. VP → V NP
d. V → likes
e. NP → John
f. NP → Lyn

String rewriting derivation

1. S $\rightarrow$ **NP** VP (1a)
2. $\rightarrow$ John **VP** (1e)
3. $\rightarrow$ John really **VP** (1b)
4. $\rightarrow$ John really **V** NP (1c)
5. $\rightarrow$ John really likes **NP** (1d)
6. $\rightarrow$ John really likes Lyn (1f)

Tree Substitution Grammars

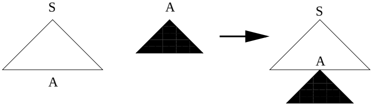
- Elementary structures are trees
- A down arrow (↓) indicates where a substitution takes place



Substitution operation

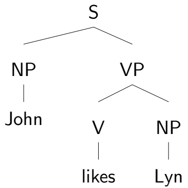
The substitution operation allows us to insert elementary trees into other elementary trees

- Where there is a (non-terminal) node marked for substitution (↓) on the **frontier**, an elementary tree **rooted** in the same category can be substituted there



Final tree

So, we end up with the following **derived tree**



Notes:

- order of substitutions is irrelevant
- This tree is **completed** = there are no substitution nodes left on the frontier

Elementary trees

Let's step back a little and look at the building blocks of TAG. Our basic elements are **elementary trees**, which come in two guises:

- **initial trees**, which have:
  - root node
  - interior nodes labeled by non-terminal symbols
  - frontier nodes of terminal and non-terminal symbols; substitution nodes are marked by the down arrow (↓)

  ⇒ Tree Substitution Grammars (TSGs) only use initial trees

Elementary trees (cont.)

- **auxiliary trees**, which have
  - root node
  - interior nodes labeled by non-terminal symbols
  - frontier nodes similar to usage in initial trees, but with a designated (*) **foot node** = identical label to the root node

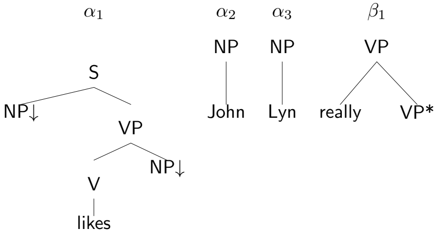
  ⇒ TAGs need auxiliary trees for adjunction

⇒ In LTAG, at least one frontier node must be a terminal symbol (lexical item)

Lexicalization

Lexicalization is the process of associating at least one terminal element with every elementary tree.

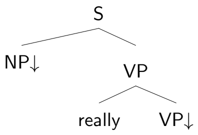
Adjunction is necessary if we want to lexicalize the grammars in a linguistically meaningful way, i.e., substitution isn't enough.



The need for adjunction

With the elementary trees above and using only substitution, there is no way to generate *John really likes Lyn*.

We would need an elementary tree along the following, unappealing lines:



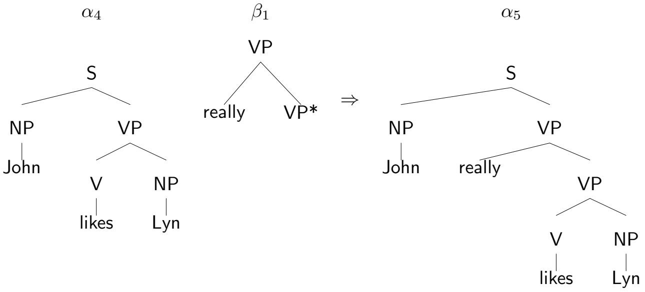
Adjunction

So, we introduce the **adjunction** operation, which is where auxiliary trees come in.

- We can now insert one tree into another, provided that the nodes match up
- That is, an auxiliary tree can modify an XP iff its root and foot nodes are both labeled XP

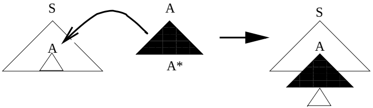
Using adjunction and substitution gives us true **Tree Adjoining Grammars (TAGs)**

Adjunction example



Adjunction operation

- An auxiliary tree is inserted into an initial tree (or derived tree) by cutting the initial/derived tree into two parts, above and below a node (A)
  - The node of the root of the auxiliary tree is identified with the node A
  - The node of the foot of the auxiliary tree is identified with the root of the excised tree



Adjunction (Adjoining) Constraints

Adjunction sometimes needs to be constrained even more than by ensuring category identity

- **Selective Adjunction** (SA(T)): only members of T, a set of auxiliary trees, may adjoin at this node
- **Null Adjunction** (NA): no adjunction is allowed at this node
- **Obligatory Adjunction** (OA(T)): a member of T must adjoin at this node

Overview

Derived Trees and Derivation Trees

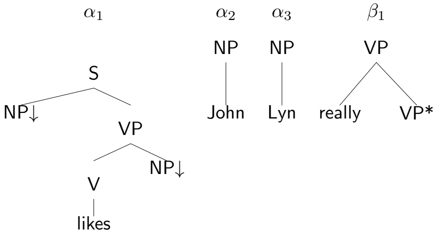
TAG distinguishes between **derived trees** and **derivation trees**.

- Dervied trees are akin to context-free/phrase structure trees
- Derivation trees are akin to dependency trees

TAG provides a way of having both kinds of representations

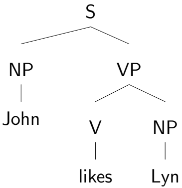
Example Lexicon

Recall the following lexical entries:



Derived Tree

The derived tree is obtained by gluing all the tree pieces together until there's a normal-looking PS tree:



But this tells us nothing about how the tree was derived.

Derivation Trees

The derivation tree records a history of the derviation and in the process captures the dependency relations among words in the sentence

$$\alpha_{like}$$

$$\alpha_{John}(1) \qquad \alpha_{Lyn}(2.2) \qquad \beta_{really}(2)$$

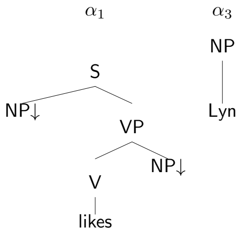
How to come up with a derivation tree

Each node in the derivation tree records the address of the node in the parent tree to which the adjunction/substitution was performed

- 0 is the root node address
- k is the address of the k^{th} child of the root node
- $p.q$ is the address of the q^{th} child of the node at address p (sort of like the q^{th} child of the p^{th} child)

Derivation tree address

Lyn gets the annotation 2.2 because VP is the second daughter of S, and NP is the second daughter of VP



TAG: summary

- MCS formalism
- Lexicalized
- Can be made parsable
- Derived and derivation trees allow for a way to deal with syntax and semantics
- Can be equipped with semantics (synchronous tag)

Overview

Motivation for LFG

- *Lexical* = (not transformational) richly structured lexicon, where relations between, e.g., verbal alternations, are stated
- *Functional* = (not configurational) abstract grammatical functions like subject and object are primitives, i.e., not defined by the phrase structure configurations

LFG in a nutshell

LFG (minimally) distinguishes two kinds of representation:

- *c-structure* (constituent structure):
  overt linear and hierarchical organization of words into phrases
- *f-structure* (functional structure):
  abstract functional organization of the sentence, explicitly representing syntactic predicate-argument structure and functional relations

These are two completely different formalisms: trees (c-structure) and attribute-value matrices (f-structure)

(We will largely ignore A-structure and σ-structure here.)

Overview

Part I: F-structure

F-structure maps more closely to meaning and encodes abstract grammatical relations like subject and object as *primitives*, i.e. not reducible to anything else (e.g., tree structure)

Motivation:

- Study of grammatical relations predates modern linguistic theory
- Categories like subject and object are cross-linguistic $\rightarrow$ languages vary less in their f-structure
- e.g., Keenan-Comrie Hierarchy (for relative clause formation) is supposedly universal

  $\text{SUBJ} > \text{DO} > \text{IO} > \text{OBL} > \text{GEN} > \text{OCOMP}$

Grammatical functions

Inventory: SUBJect, OBJect, OBJ_θ, COMP, XCOMP, OBLique_θ, ADJunct, XADJunct

- Terms (core functions): SUBJ, OBJ, OBJ_θ
- Semantically restricted: OBJ_θ, OBL_θ
  - Thematic restrictions (θ) placed on function
  - OBJ_θ: secondary OBJ functions associated with thematic roles: $\text{OBJ}_{\text{THEME}}$ only one used in English
  - OBL_θ: thematically restricted oblique functions, often corresponding to adpositions
- Open clausal functions (no internal subject): XCOMP, XADJ
  - COMP: sentential or closed (nonpredicative) infinitival complement
  - XCOMP: open (predicative) complement with subject externally controlled

Subcategorization

Subcategorization is done at f-structure

- Verbs select for grammatical functions
- Use the PRED (predicate) feature to specify the semantic form, e.g.,
  - *yawn*: PRED 'YAWN<SUBJ>'
  - *hit*: PRED 'HIT<SUBJ,OBJ>'
  - *give*: PRED 'GIVE<SUBJ,OBJ,$\text{OBJ}_{\text{THEME}}$>'
  - *eat*: PRED 'EAT<SUBJ,(OBJ)>'

F-structure representation: Simple F-structures

F-structure is a function from attributes to values

- For the proper noun *David*, PRED and NUM are attributes; 'DAVID' and SG are the corresponding values

(1) $\begin{bmatrix} \text{PRED} & \text{'DAVID'} \\ \text{NUM} & \text{SG} \end{bmatrix}$

- F-structures within f-structures: *David yawned*

(2) $\begin{bmatrix} \text{PRED} & \text{'YAWN<SUBJ>'} \\ \text{TENSE} & \text{PAST} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'DAVID'} \\ \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$

F-structure features

What sorts of features can be used?

- Ultimately, that's up to the grammar writer
- Commonly used features in LFG include ASPECT, PRONTYPE, VFORM, etc. (see (17) in Dalrymple (2006))

Important note:

- LFG does not define a set of features or values which *must* be included in an f-structure
- So, one verb may define VFORM, while another might leave it undefined.
  - This is different from HPSG, as we'll see.

F-structures

- make use of a **unification** paradigm

Unification Operation

$$\begin{bmatrix} \textit{Agreement} & [\textit{Number sg}] & \\ \textit{Subject} & [\textit{Agreement} & [\textit{Number sg}]] \end{bmatrix}$$

$$U$$

$$[\textit{Subject} \quad [\textit{Agreement} \quad [\textit{Person 3}]]]$$

$$=$$

$$\begin{bmatrix} \textit{Agreement} & [\textit{Number sg}] \\ \textit{Subject} & \begin{bmatrix} \textit{Agreement} & \begin{bmatrix} \textit{Number} & \textit{sg} \\ \textit{Person} & \textit{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- are associated with c-structure nodes,
- described mostly by **equations**

Functional constraints example

Lexical constraints:

- *John*
  - $(g\ \text{PRED}) = \text{'JOHN'}$
  - $(g\ \text{NUM}) = \text{SG}$
- *runs*
  - $(f\ \text{PRED}) = \text{'RUN<SUBJ>'}$
  - $(f\ \text{SUBJ CASE}) = \text{NOM}$
  - $(f\ \text{SUBJ NUM}) = \text{SG}$

Phrasal constraints (more on this later):

- $(f\ \text{SUBJ}) = g$

Functional constraints example (cont.)

Combining lexical and phrasal constraints, we have:

- $(f \text{ SUBJ}) = g$
- $(g \text{ PRED}) = \text{'J}\textsc{ohn}\text{'}$
- $(g \text{ NUM}) = \text{SG}$
- $(f \text{ PRED}) = \text{'RUN<SUBJ>'}$
- $(g \text{ CASE}) = \text{NOM}$
- $(g \text{ NUM}) = \text{SG}$

Minimal solution:

$$f:\begin{bmatrix} \text{PRED} & \text{'RUN<SUBJ>'} \\ \text{SUBJ} & g:\begin{bmatrix} \text{PRED} & \text{'J}\textsc{ohn}\text{'} \\ \text{CASE} & \text{NOM} \\ \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

Overview

C-Structure

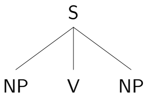
- Corresponds to a fairly traditional notion of phrase structure
- X-Bar Theory (lexical heads with specifiers and complements)
- Adjunction operation XP $\rightarrow$ XP YP
- Categories: lexical (N, P, V, A, Adv) and functional (I, C)
  no fixed inventory
- Optionality: all constituent structure positions are optional
- A grammar can also use
  - metacategories
  - ID/LP rules

Metacategories

A **metacategory** represents several different sets of categories

(18) a. XP $\equiv$ {NP | PP | VP | AP | AdvP}
b. VP $\equiv$ V NP

Note that using the metacategory VP given in (18b) in the rule S $\rightarrow$ NP VP results in the following tree:



ID/LP Rules

Rules can be written in ID/LP format: ID = immediate dominance, LP = linear precdence

(19) No LP rules:

a. VP $\rightarrow$ V, NP

b. VP $\rightarrow$ {V NP | NP V}

(20) One LP rule:

a. VP $\rightarrow$ V, NP $\qquad$ V $<$ NP

b. VP $\rightarrow$ V NP

(21) Interacting LP rules:

a. VP $\rightarrow$ V, NP, PP $\qquad$ V $<$ NP, V $<$ PP

b. VP $\rightarrow$ {V NP PP | V PP NP}

Overview

An example grammar: The c-structure rules with annotations

(based on Kaplan and Bresnan 1995)

(27) a.
$$\mathrm{S} \rightarrow \begin{array}{cc} \mathrm{NP} & \mathrm{VP} \\ (\uparrow \textsc{subj}) = \downarrow & \uparrow = \downarrow \end{array}$$

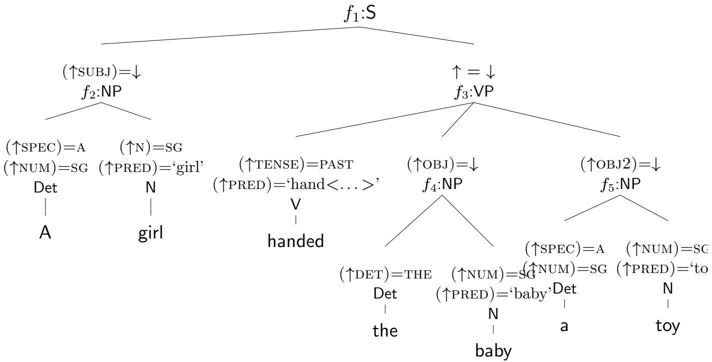
b.
$$\mathrm{NP} \rightarrow \begin{array}{cc} \mathrm{Det} & \mathrm{N} \\ \uparrow = \downarrow & \uparrow = \downarrow \end{array}$$

c.
$$\mathrm{VP} \rightarrow \begin{array}{ccc} \mathrm{V} & \mathrm{NP} & \mathrm{NP} \\ \uparrow = \downarrow & (\uparrow \textsc{obj}) = \downarrow & (\uparrow \textsc{obj2}) = \downarrow \end{array}$$

An example grammar II: The lexicon

(28)	a.	*a*	Det	$(\uparrow\text{SPEC}) = \text{A}$ $(\uparrow\text{NUM}) = \text{SG}$
	b.	*girl*	N	$(\uparrow\text{NUM}) = \text{SG}$ $(\uparrow\text{PRED}) =$ 'girl'
	c.	*handed*	V	$(\uparrow\text{TENSE}) = \text{PAST}$ $(\uparrow\text{PRED}) =$ 'hand$<(\uparrow\text{SUBJ}), (\uparrow\text{OBJ}), (\uparrow\text{OBJ2})>$'
	d.	*the*	Det	$(\uparrow\text{SPEC}) = \text{THE}$
	e.	*baby*	N	$(\uparrow\text{NUM}) = \text{SG}$ $(\uparrow\text{PRED}) =$ 'baby'
	f.	*toy*	N	$(\uparrow\text{NUM}) = \text{SG}$ $(\uparrow\text{PRED}) =$ 'toy'

A sentence licensed by the example grammar



The resulting f-structure for the example sentence

$$
f_1, f_3 \colon \left[\begin{array}{ll}
\text{SUBJ} & f_2 \colon \left[\begin{array}{ll} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{'girl'} \end{array}\right] \\
\text{TENSE} & \text{PAST} \\
\text{PRED} & \text{'hand} < (\uparrow\text{SUBJ}), (\uparrow\text{OBJ}), (\uparrow\text{OBJ2}) > \text{'} \\
\text{OBJ} & f_4 \colon \left[\begin{array}{ll} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{'baby'} \end{array}\right] \\
\text{OBJ2} & f_5 : \left[\begin{array}{ll} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{'toy'} \end{array}\right]
\end{array}\right]
$$

Summary

- LFG is split into f-structure and c-structure, with a mapping between them
- F-structure is a rich feature-based way of encoding functional relations
- C-structure is a basic constituent structure

Overview

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 (GPSG) (Gazdar et al. 1985)

2

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

3

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)

4

Skeleton of a typed feature structure

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

$$\begin{bmatrix} \textit{sign} & \\ \text{PHON} & \textit{list}\left(\text{PHON}\right) \\ \text{SYNSEM} & \begin{bmatrix} \textit{synsem} & \\ \text{LOCAL} & \textit{local} \\ \text{NON-LOCAL} & \textit{non-local} \end{bmatrix} \\ \text{DTRS} & \textit{list}\left(\text{SIGN}\right) \end{bmatrix}$$

5

Abbreviated skeleton

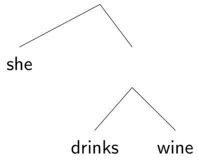
Things are often abbreviated when written down (although, the object itself still contains the same things):

$$\begin{bmatrix} \text{PHON} & \textit{list}\left(\text{PHON}\right) \\ \text{SYNSEM} & \begin{bmatrix} \text{LOC} & \textit{loc} \\ \text{NON-LOC} & \textit{non-loc} \end{bmatrix} \\ \text{DTRS} & \textit{list}\left(\text{SIGN}\right) \end{bmatrix}$$

6

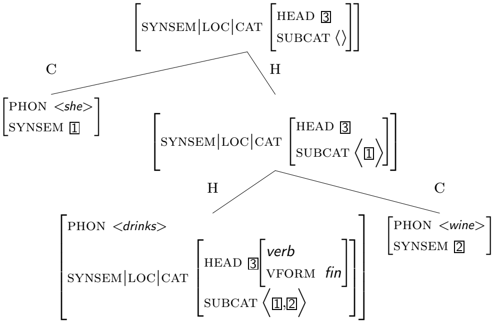
An example tree

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



7

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 ($\boxed{3}$)
- We uses lists to encode subcategorization information, and these items are identified with elements in the tree—note, too, how selection is kept local

9

Phrase structure grammar?

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

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{S|L|C} & \begin{bmatrix}\text{HEAD} & \boxed{3}\\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\\
\text{DTRS} & \begin{bmatrix}
\textit{head-comps-struc} & \\
\text{COMP-DTRS} & \left\langle \begin{bmatrix}\text{PHON} & <\textit{she}>\\ \text{SYNSEM} & \boxed{1}\end{bmatrix}\right\rangle\\
\text{HEAD-DTR} & \begin{bmatrix}
\textit{phrase} & \\
\text{S|L|C} & \begin{bmatrix}\text{HEAD} & \boxed{3}\\ \text{SUBCAT} & \langle\boxed{1}\rangle\end{bmatrix}\\
\text{DTRS} & \begin{bmatrix}
\textit{head-comps-struc} & \\
\text{HEAD-DTR} & \begin{bmatrix}
\textit{word} & \\
\text{PHON} & <\textit{drinks}>\\
\text{S|L|C} & \begin{bmatrix}\text{HEAD} & \boxed{3}\begin{bmatrix}\textit{verb} & \\ \text{VFORM} & \textit{fin}\end{bmatrix}\\ \text{SUBCAT} & \langle\boxed{1},\boxed{2}\rangle\end{bmatrix}
\end{bmatrix}\\
\text{COMP-DTRS} & \left\langle\begin{bmatrix}\text{PHON} & <\textit{wine}>\\ \text{SYNSEM} & \boxed{2}\end{bmatrix}\right\rangle
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

11

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)

12

Lexical entry for *put*

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & <\textit{put}> \\ \text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CATEGORY} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{AUX} & - \end{bmatrix} \\ \text{VALENCE} & \begin{bmatrix} \text{SUBJ} & \left\langle \text{NP}_{\boxed{1}} \right\rangle \\ \text{COMPS} & \left\langle \text{NP}_{\boxed{2}}, \text{PP}_{\boxed{3}} \right\rangle \end{bmatrix} \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \textit{put-relation} & \\ \text{PUTTER} & \boxed{1} \\ \text{THING-PUT} & \boxed{2} \\ \text{DESTINATION} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

13

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

14

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

$$
\begin{bmatrix}
\textit{word} & \\
\text{PHON} & <\textit{seem}> \\
\text{SYNSEM}|\text{LOCAL} & \begin{bmatrix}
\text{CAT}|\text{VAL} & \begin{bmatrix}
\text{SUBJ} & \boxed{1} \\
\text{COMPS} & \left\langle \begin{bmatrix}
\text{LOC} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & \textit{verb} \\
\text{VAL} & \begin{bmatrix}
\text{SUBJ} & \boxed{1} \\
\text{COMPS} & \langle\rangle
\end{bmatrix}
\end{bmatrix} \\
\text{CONT} & \boxed{2}
\end{bmatrix}
\end{bmatrix} \right\rangle
\end{bmatrix} \\
\text{CONTENT} & \begin{bmatrix}
\textit{seem'} & \\
\text{ARG} & \boxed{2}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

16

Control/Equi verb example

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle\textit{try}\rangle \\ \text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CAT}|\text{VAL} & \begin{bmatrix} \text{SUBJ} & \left\langle \text{NP}_{\boxed{1}} \right\rangle \\ \text{COMPS} & \left\langle \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{VAL} & \begin{bmatrix} \text{SUBJ} & \left\langle \text{NP}_{\boxed{1}} \right\rangle \\ \text{COMPS} & \langle\rangle \end{bmatrix} \end{bmatrix} \\ \text{CONT} & \boxed{2} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \textit{try'} & \\ \text{TRYER} & \boxed{1} \\ \text{TRIED} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

17

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

18

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.

19

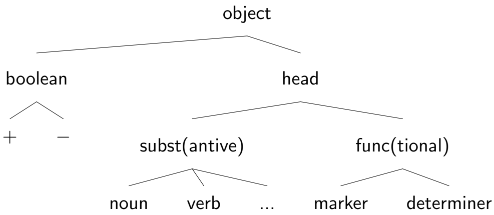
The signature

- 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

20

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 attribute has an appropriate value.
    - Note that this is different from LFG.
  - *sort-resolved*: Every node is of a maximally specific type.

23

Structure sharing

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

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & <\textit{walks}> \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT}|\text{SUBCAT} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{3rd,sing}] \right\rangle \\ \text{CONTENT} & \begin{bmatrix} \textit{walk'} & \\ \text{WALKER} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

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

24

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., $\left[\text{SPOUSE}\ \left[\text{NAME}\ \textit{mary}\right]\right]$
- **Tags** (structure sharing) to specify **token identity**, e.g. $\boxed{1}$

25

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.

26

Description example

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

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

(3) On the verb's SUBJ list: $\begin{bmatrix} \textit{word} & \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT}|\text{HEAD} & \textit{noun} \\ \text{CONTENT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{NUM} & \textit{sing} \\ \text{GEN} & \textit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

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

27

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.

28

Subsumption example

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

(4) a.
$$\begin{bmatrix} \textit{word} & \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT}|\text{HEAD} & \textit{noun} \\ \text{CONTENT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{PER} & \textit{1st} \\ \text{NUM} & \textit{sing} \\ \text{GEN} & \textit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{word} & \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT}|\text{HEAD} & \textit{noun} \\ \text{CONTENT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{PER} & \textit{3rd} \\ \text{NUM} & \textit{sing} \\ \text{GEN} & \textit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

29

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

30

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

$$\begin{bmatrix} \textit{sign} & \\ \textsc{phon} & \textit{list(phonstring)} \\ \textsc{synsem} & \textit{synsem} \end{bmatrix}$$

word $\qquad \begin{bmatrix} \textit{phrase} & \\ \textsc{dtrs} & \textit{constituent-structure} \end{bmatrix}$

$$\begin{bmatrix} \textit{synsem} & \\ \textsc{local} & \textit{local} \\ \textsc{non-local} & \textit{non-local} \end{bmatrix} \qquad \begin{bmatrix} \textit{local} & \\ \textsc{category} & \textit{category} \\ \textsc{content} & \textit{content} \\ \textsc{context} & \textit{context} \end{bmatrix} \qquad \begin{bmatrix} \textit{category} & \\ \textsc{head} & \textit{head} \\ \textsc{subcat} & \textit{list(synsem)} \\ \ldots & \end{bmatrix}$$

Why the complicated structure?

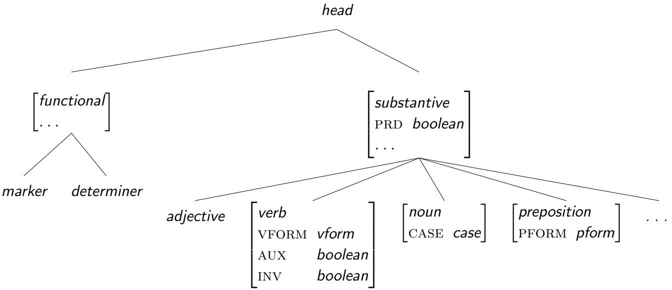
- LOCAL & NONLOCAL: Most linguistic constructions can be handled locally, but non-local constructions (e.g., extraction) require different mechanisms

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

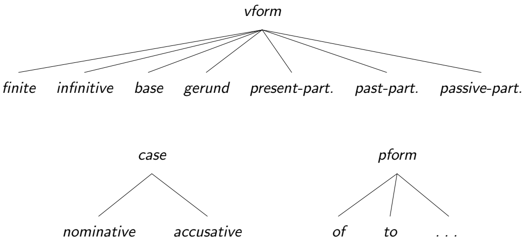
- HEAD and SUBCAT: a words syntactic information comes in two parts: its own lexical information (part of speech, etc.) and information about its arguments

33

Part-of-speech (HEAD information)



Properties of particular part-of-speech



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

36

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.

⇒ Constructions are thus restricted to local relations

37

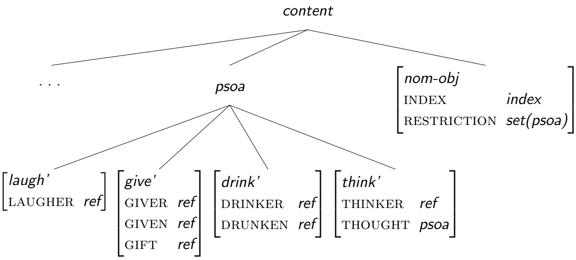
CONTENT information

The CONTENT feature specifies different semantic information

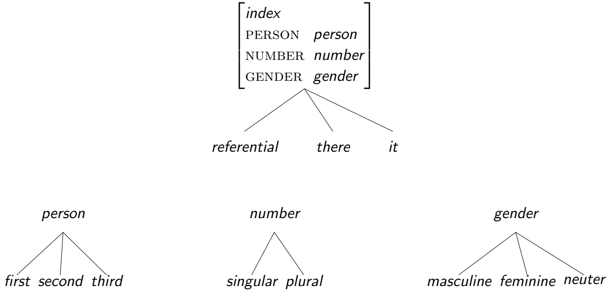
- A feature appropriate for *nominal-object* objects (a subtype of *content* objects) is INDEX (as shown on the next slide)
- Agreement features can be stated through the INDEX feature
- Note that CASE was put somewhere else (within HEAD), so CASE agreement is treated differently than person, number,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

38

Semantic representations



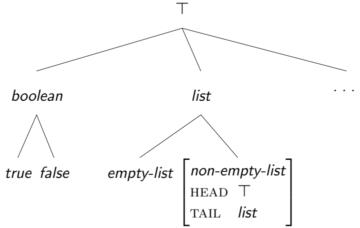
Indices


index
PERSON person
NUMBER number
GENDER gender
referential
there
it
person
first
second
third
number
singular
plural
gender
masculine
feminine
neuter

40

Auxiliary data structures

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



41

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	$\left\langle \boxed{1} \mid \boxed{2} \right\rangle$
$\left\langle \ldots \boxed{1} \mid \langle\rangle \right\rangle$	is abbreviated as	$\left\langle \ldots \boxed{1} \right\rangle$
$\begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{TAIL} & \begin{bmatrix} \text{HEAD} & \boxed{2} \\ \text{TAIL} & \boxed{3} \end{bmatrix} \end{bmatrix}$	is abbreviated as	$\left\langle \boxed{1}, \boxed{2} \mid \boxed{3} \right\rangle$

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

42

Sorbonne Nouvelle

Abbreviations of common AVMs

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

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

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.:

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & <\textit{laughs}> \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \textit{fin} \end{bmatrix} \\ \text{SUBCAT} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}}[\textit{3rd,sing}] \right\rangle \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \textit{laugh'} & \\ \text{LAUGHER} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

44

An example lexicon

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

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

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

46

$$\bigvee \begin{bmatrix} \text{PHON} & <\textit{she}> \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{noun} & \\ \text{CASE} & \textit{nom} \end{bmatrix} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{PER} & \textit{third} \\ \text{NUM} & \textit{sing} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\bigvee \begin{bmatrix} \text{PHON} & <\textit{wine}> \\ \text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{PER} & \textit{third} \\ \text{NUM} & \textit{sing} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

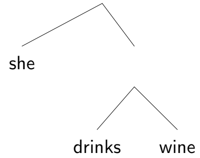
$$\bigvee \begin{bmatrix} \text{PHON} & <\textit{to}> \\ \text{S}|\text{L} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{preposition} \\ \text{PFORM} \; \textit{to} \end{bmatrix} \\ \text{SUBCAT} & \left\langle \text{NP}[\textit{acc}]_{\boxed{1}} \right\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} \; \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\bigvee \begin{bmatrix} \text{PHON} & <\textit{think}> \\ \text{S}|\text{L} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{VFORM} \; \textit{fin} \end{bmatrix} \\ \text{SUBCAT} & \left\langle \text{NP}[\textit{nom}]_{\boxed{1}[\textit{plur}]}, \; \text{S}[\textit{fin}]:\boxed{2} \right\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \textit{think'} & \\ \text{THINKER} & \boxed{1} \\ \text{THOUGHT} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

48

A very first sketch of an example

Here's that impoverished tree again:



We're going to see how the theory licenses this structure ...

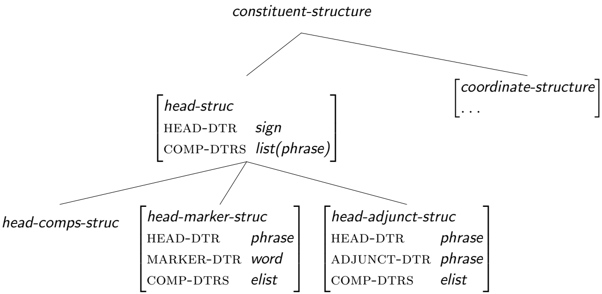
Types of phrases

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

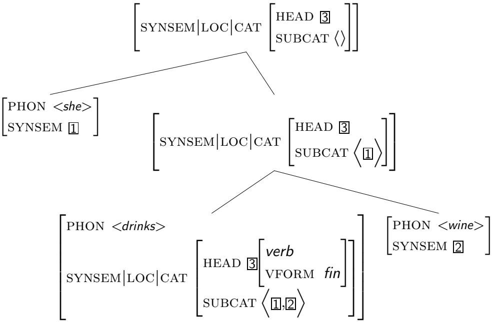
- Each *phrase* has a DTRS attribute (*words* do not have this attribute), which has a *constituent-structure* value
- This DTRS value loosely corresponds to what we normally view in a tree as daughters
  - Additionally, “tree branches” contain grammatical role information (adjunct, complement, etc.)
- By distinguishing different kinds of *constituent-structures*, we define what kinds of phrases exist in a language

50

An ontology of phrases



Sketch of an example for head-complement structures



52

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.

53

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**:

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

54

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).

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

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

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

55

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

56

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

Formally, though, these constraints are phrased as the universal principles were

57

Immediate Dominance Principle (for English):

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

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

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

$\vee$ . . . continued on next page

Immediate Dominance Principle (for English):

$$
\begin{bmatrix} \textit{phrase} \\ \text{DTRS} \;\; \textit{headed-struc} \end{bmatrix} \rightarrow \quad \begin{array}{l} \vdots \\ \vdots \\ \vdots \\ \vee \begin{bmatrix} \text{DTRS} \; \begin{bmatrix} \textit{head-marker-struc} \\ \text{MARKER-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD} \;\; \textit{marker} \end{bmatrix} \end{bmatrix} \text{(Head-Marker)} \\ \vee \begin{bmatrix} \text{DTRS} \; \begin{bmatrix} \textit{head-adjunct-struc} & \\ \text{ADJ-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD}|\text{MOD} & \boxed{1} \\ \text{HEAD-DTR}|\text{SYNSEM} & \boxed{1} \end{bmatrix} \end{bmatrix} \quad \text{(Head-Adjunct)} \end{array}
$$

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*

59

Towards Head Adjunct Structures

Lexical entry of an attributive adjective

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{word}} \\
\text{PHON} & \langle \textit{red} \rangle \\
\text{SYNSEM} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{synsem}} \\
\text{LOCAL} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{local}} \\
\text{CATEGORY} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{cat}} \\
\text{HEAD} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{adj}} \\
\text{PRD} & - \\
\text{MOD} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{synsem}} \\
\text{LOCAL} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{local}} \\
\text{CATEGORY} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{cat}} \\
\text{HEAD} & \textit{noun} \\
\text{SUBCAT} & \left\langle \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{synsem}} \\
\text{LOCAL} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{local}} \\
\text{CATEGORY} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{cat}} \\
\text{HEAD} & \textit{det}
\end{array}\right]
\end{array}\right]
\end{array}\right] \right\rangle
\end{array}\right] \\
\text{CONTENT} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{nom-obj}} \\
\text{INDEX} & \boxed{1}\textit{index} \\
\text{RESTR} & \boxed{2}\textit{list}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{CONTENT} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{nom-obj}} \\
\text{INDEX} & \boxed{1} \\
\text{RESTR} & \left\langle \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{red-rel}} \\
\text{ARG1} & \boxed{1}
\end{array}\right] \middle| \; \boxed{2} \right\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

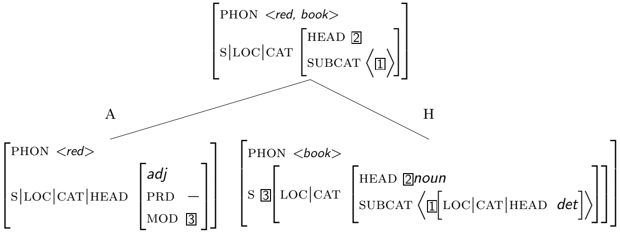
60

Lexical entry of an attributive adjective
Version without redundant specifications

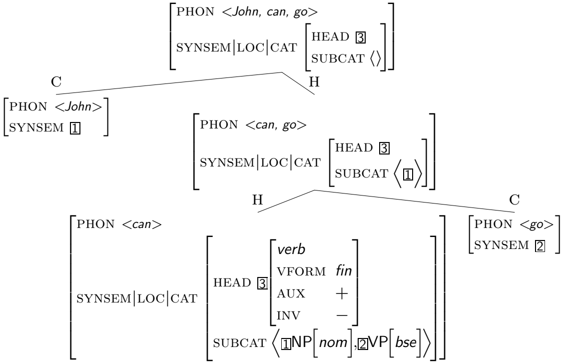
$$
\begin{bmatrix}
\textit{word} & \\
\text{PHON} & <\textit{red}> \\
\text{SYNSEM}|\text{LOC} & \begin{bmatrix} \text{CAT}|\text{HEAD} & \begin{bmatrix} \textit{adj} & \\ \text{PRD} & - \\ \text{MOD}|\text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \left\langle \begin{bmatrix} \text{LOC}|\text{CAT}|\text{HEAD} & \textit{det} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \text{INDEX} & \boxed{1}\textit{index} \\ \text{RESTR} & \boxed{2}\textit{list} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{1} \\ \text{RESTR} & \left\langle \begin{bmatrix} \textit{red-rel} & \\ \text{ARG1} & \boxed{1} \end{bmatrix} \middle| \boxed{2} \right\rangle \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

61

Sketch of an example for a head-adjunct structure

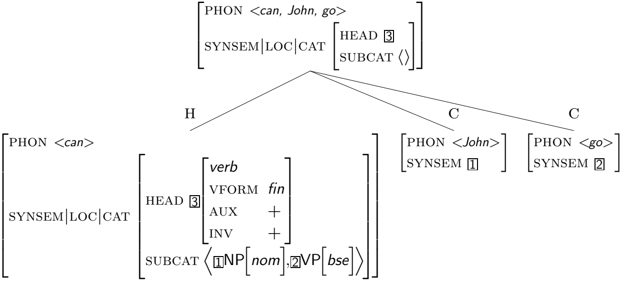


Sketch of an example with an auxiliary



63

Sketch of an example with an inverted auxiliary



64

SPEC Principle:

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

$$\rightarrow \begin{bmatrix} \text{DTRS} & \begin{bmatrix} \left(\text{MARKER-DTR} \lor \text{COMP-DTRS}|\text{FIRST}\right)|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD}|\text{SPEC} & \boxed{1} \\ \text{HEAD-DTR}|\text{SYNSEM} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

65

Marking Principle:

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

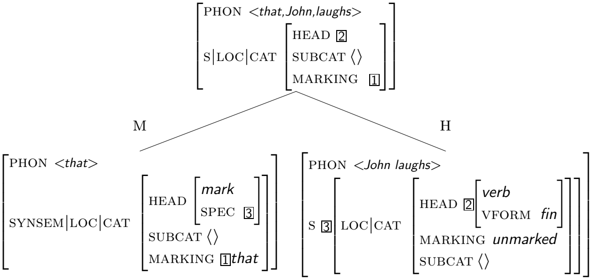
66

Lexical entry of the marker *that*

$$\begin{bmatrix} \text{PHON} & <\textit{that}> \\ \text{SYNSEM}|\text{LOC}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{mark} & \\ \text{SPEC} & \begin{bmatrix} \text{LOC}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \textit{fin} \vee \textit{bse} \end{bmatrix} \\ \text{MARKING} & \textit{unmarked} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{SUBCAT} & \langle\rangle \\ \text{MARKING} & \textit{that} \end{bmatrix} \end{bmatrix}$$

67

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

References I

Dickinson, M., Brew, C., & Meurers, D. 2012. *Language and Computers*. Wiley.