Foundations of Language Science and Technology

Lecture 3: Linguistic Foundations I (02.11.2005)

Valia Kordoni

Email: kordoni@coli.uni-sb.de

Dependency Grammar (DG)

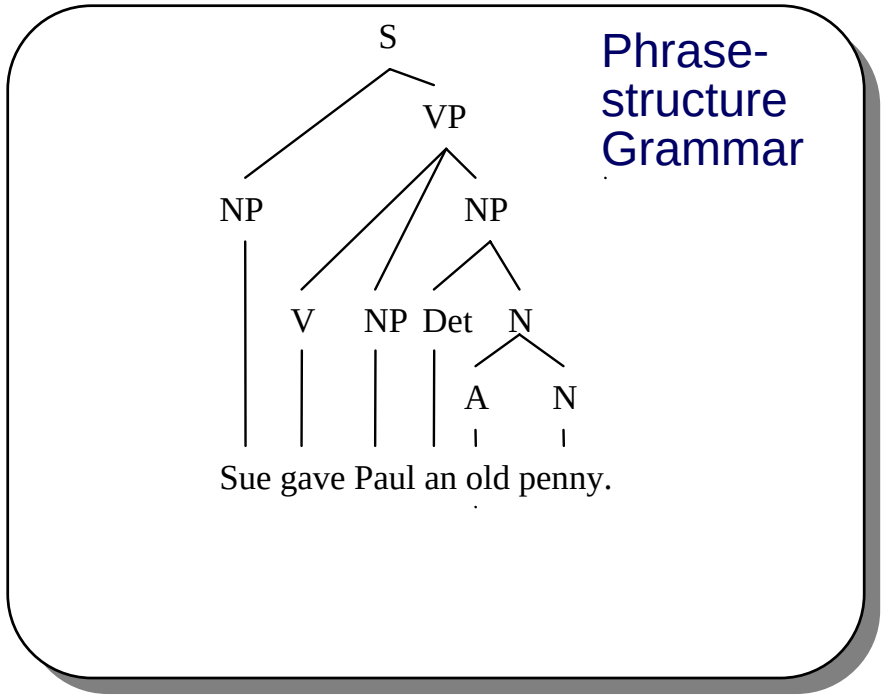
- The way to analyse a sentence is by looking at the relations between words
- A verb and its valents/arguments drive an analysis, which is closely related to the semantics of a sentence
- No grouping, or constituency, is used

Phrase Structure Grammars (PSGs)

- Grouping, or constituency, is used

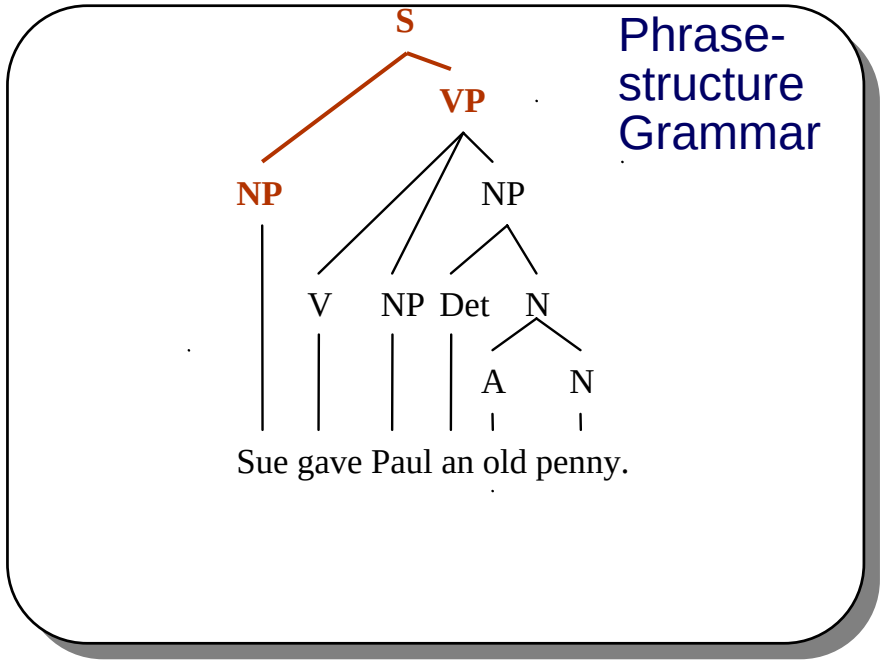
(1) Sue gave Paul an old penny.

Phrase Structure Grammars (PSGs)



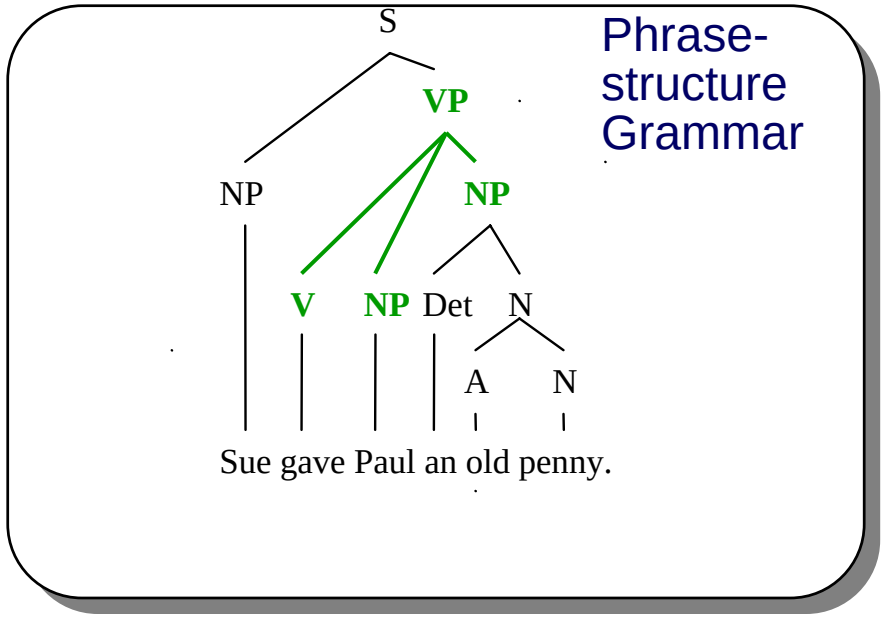
$S \rightarrow NP\ VP$

Phrase Structure Grammars (PSGs)



$S \rightarrow NP\ VP$

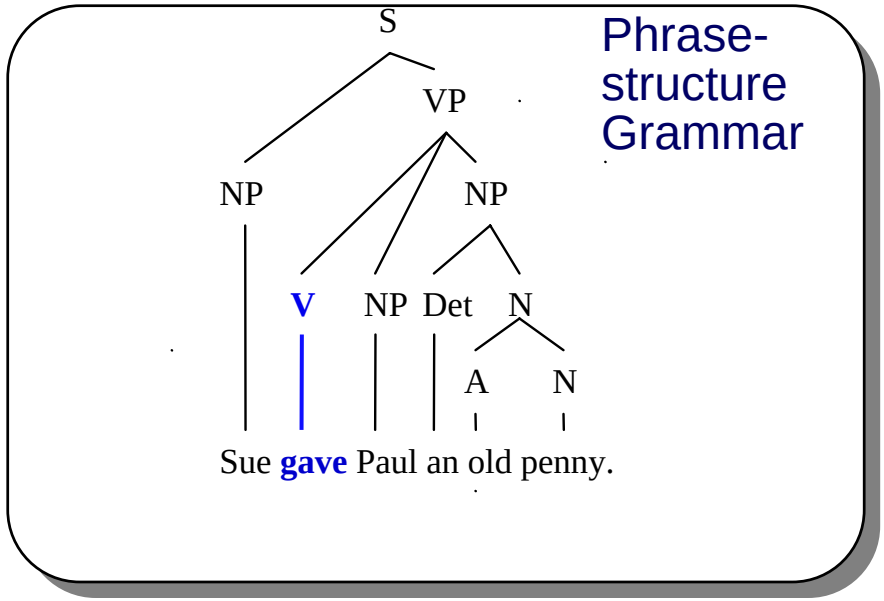
Phrase Structure Grammars (PSGs)



S → NP VP

VP → V NP NP

Phrase Structure Grammars (PSGs)



$S \rightarrow NP\ VP$
$VP \rightarrow V\ NP\ NP$

$V \rightarrow$ gave

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. Colourless green ideas sleep furiously.
  2. *Furiously sleep ideas green colourless.
- What generalisations are needed to capture the difference between grammatical and ungrammatical sentences?

Syntax: What does it mean?

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

- Psychological way/model: syntactic structures correspond to what is in heads of speakers and hearers
- Computational way/model: syntactic structures are formal objects which can be mathematically manipulated

The Transformational Tradition

Roughly speaking, **transformational syntax** (GB = Government and Binding, P&P = Principles and Parameters,...) 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?
- Theory-driven: data should ideally fit with a theory already in place (often based on English)

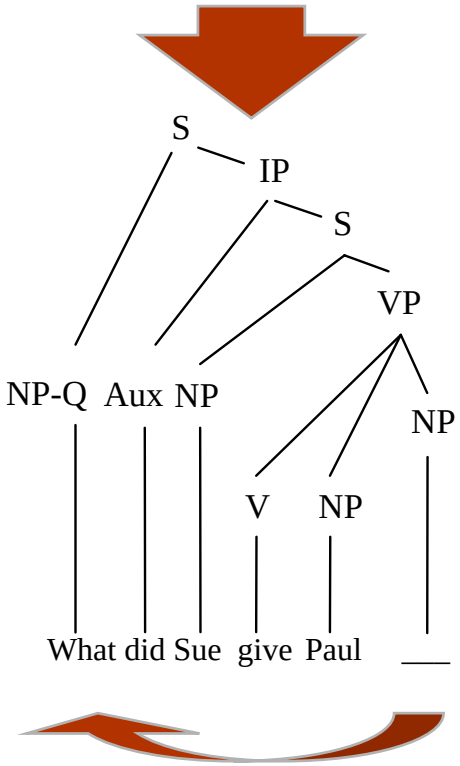
The Transformational Tradition (cont.)

- Universality: generalisations must be applicable to all languages
- Transformations: (surface) sentences are derived from underlying other sentences, e.g., passives are derived from active sentences

But this kind of theory does not lend itself well to computational applications

The Transformational Tradition (cont.)

Sue gave Paul an old penny



Making it computational

How is a grammatical theory useful for computational linguistics?

- 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

=> Both tasks are often subparts of practical applications, such as dialogue systems, for instance

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 one wants 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 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

Computational Grammar formalisms share several properties:

- Descriptive adequacy
- Precise encodings (implementable)
- Constrained mathematical formalism
- Monostratalism
- (Usually) high lexicalism

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
- Gives us an accurate encoding of a language
- Gives us broad-coverage, i.e., can (try to) describe all of a language

  → No notion of core and periphery phenomena

Precise Encodings

Mathematical Formalism: formal way to generate sets of strings

Precisely define:

- elementary structures
- ways of combining those structures

=> Such an emphasis on mathematical precision makes these grammar formalisms more easily implementable

Constrained Mathematical Formalism

A formalism must be **constrained**, i.e., it 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

Monostratal Frameworks

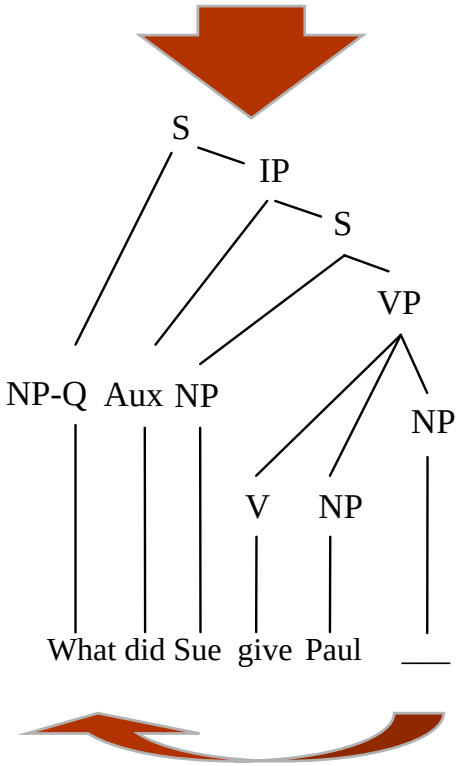
Only have one (surface) syntactic level

- Make no recourse to movement
- Augment your basic (phrase structure) tree with information that can describe „movement“ phenomena

=> Without having to refer to movement, easier to process sentences on a computer

This should be avoided!

Sue gave Paul an old penny



Lexical

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

- Linguistic motivation: lexicon is the best way to specify some generalisations: *He told/*divulged me the truth*
- Computational motivation: can derive lexical information from corpora (large computer-readable texts)

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

Context-Free Grammars (CFGs)

Context-Free Grammars (CFGs) are one kind of constrained mathematical formalism, a precise way of encoding syntactic rules:

- elementary structures: rules composed of non-terminal and terminal elements
- combine rules by rewriting them

Context-Free Rules

Example of a set of rules:

- S → NP VP
- NP → Det N
- VP → V NP
- ...

But these rules are rather impoverished.

Are CFGs good enough?

- Data from various languages show that CFGs are not powerful enough to handle all natural language constructions
- CFGs are not easily lexicalised
- CFGs become complicated once we start taking into account agreement features, verb subcategorisations, unbounded dependency constructions, raising constructions, etc.

We need more refined formalisms...

Beyond CFGs

Move beyond CFGs, but stay „mathematical“:

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

Computational Grammar Frameworks

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

Dependency Grammar (DG)

- The way to analyse a sentence is by looking at the relations between words
- A verb and its valents/arguments drive an analysis, which is closely related to the semantics of a sentence
- No grouping, or constituency, is used

Tree-Adjoining Grammar (TAG)

- Elementary structures are trees of arbitrary height
- Trees are rooted in lexical items, i.e., lexicalised
- Put trees together by substituting and adjoining them, resulting in a final tree which looks like a CFG-derived tree

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
- CCG has sophisticated mechanisms that deal nicely with coordination, extraction, and other constructions

Lexical Functional Grammar (LFG)

- Functional structure (subject, object, etc.) divided from constituent structure (tree structure)
  - kind of like combining dependency structure with phrase structure
- Can express some generalisations 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
- Similar to LFG in its use of feature architecture
- Uses an inheritance hierarchy to relate different – types of objects (e.g., nouns and determiners are both types of nominal)

Head-driven Phrase Structure Grammar (HPSG)

- Head-Driven Phrase Structure Grammar was developed in the mid-1980s by Carl Pollard and Ivan Sag

    1. HPSG1: Pollard and Sag 1987 -- Formalism (typed feature structures), subcategorization, LP rules, the hierarchical lexicon

    2. HPSG2: Pollard and Sag 1994, Chapters 1-8 -- The structure of the sign, Control Theory, Binding Theory

Head-driven Phrase Structure Grammar (HPSG)

3. HPSG2: Pollard and Sag 1994, Chapter 9 „Reflections and Revisions“ Valence features SUBJ, COMPS, SPR
4. HPSG3: Sag, Wasow and Bender 2003

Further Developments in HPSG

- Unbounded Dependency Constructions (Sag 1997; Bouma, Malouf and Sag 2001)
- Linking Theory (Wechsler 1995; Davis 2001; Kordoni 1999, 2001)
- Semantic representation (Copestake, Flickinger, Sag and Pollard 1999)
- Argument Realization (Sag and Miller 1997)

HPSG and its influences

- Head-Driven Phrase Structure Grammar has been influenced by contemporary theories:

❖ Syntax

1. Generalized Phrase Structure Grammar (GPSG; Gazdar, Klein,Pullum and Sag 1985)
2. Categorial Grammar (CG; Mc Gee Wood 1993)
3. Lexical Functional Grammar (LFG; Kaplan and Bresnan 1982)
4. Construction Grammar (Goldberg 1995)
5. Government and Binding Theory (GB; Haegeman 1994)

HPSG and its influences (cont.)

- Head-Driven Phrase Structure Grammar has been influenced by contemporary theories:

❖ Semantics

1. Situation Semantics (Barwise and Perry 1983)
2. Discourse Representation Theory (DRT; Kamp and Reule 1993)

Key Properties of HPSG: HPSG vs. „Classical“ PSGs

- **Similarities:**

1. Both are monostratal: Every sentence is associated with a single tree structure.

2. Grammar rules have local scope only: Grammatical statements can only refer to a local tree (one node and its daughters). A tree is well-formed if and only if all its local trees are (this is true of context-free PSG only)

Key Properties of HPSG: HPSG vs. „Classical“ PSGs

- **Differences:**

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

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

   In addition, HPSG proposes several universal principles to further constrain the set of local trees admitted by the ID schemata.
3. HPSG analyses include also semantic representations.

Key Properties of HPSG: HPSG vs. Transformational Grammar

- Transformational Grammar: „Chomskyan“ frameworks, most recently formalized as Government and Binding Theory, Principles and Parameters, and Minimalism Program.

- **Similarities:**

1. Both try to account for a similar range of data.
2. Some analyses in HPSG – e.g., Binding Theory – are heavily influenced by earlier proposals in TG.
3. Both are theories of generative grammar.

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

Key Properties of HPSG: HPSG vs. Transformational Grammar

- **Differences:**

1. HPSG is non-derivational. TG is derivational.

   TG analyses start with a base generated tree, which is then subject to a variety of transformations – e.g., movement, deletion, re-analysis – that produce the desired surface structure. HPSG analyses generate only the surface tree. Rule ordering is impossible in HPSG because there is no notion of sequential derivation.

2. HPSG constraints are local. TG allows non-local statements.

3. Complex categories in HPSG are more complex than in TG. TG uses atomic categories carrying binary feature specifications. HPSG categories are very elaborated in comparison.

4. HPSG is more committed to precise formalization than TG.

5. HPSG is better suited to computational implementation.

Key Properties of HPSG and their consequences

- HPSG is monostratal, declarative, non-derivational: No transformations, no rule-ordering. In addition, analyses are surface-oriented, with a desire to avoid abstract structures such as traces and functional categories.
- HPSG is constraint-based: A structure is well-formed if and only if it satisfies all relevant constraints. Constraints are not violable, as in Optimality Theory, for example. Furthermore, constraints apply only to local trees, although local constraints can interact to have non-local effects.

Key Properties of HPSG and their consequences (cont.)

- HPSG is a lexicalist theory: Strong lexicalism (Scalise 1984). Word-internal structure (e.g., morphology) and phrase structure are handled separately. Phrasal rules cannot manipulate sub-parts of words. Lexically-governed processes such as valence alternations must be analyzed lexically.
- HPSG is head-driven (more on this later).
- HPSG is a unification-based framework where all linguistic objects are represented as typed feature structures.

Psycholinguistic Evidence for HPSG

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

1. Human language processing is incremental: That means that partial interpretation is generated for partial utterances. HPSG constraints can apply to partial structures as well as complete trees.
2. HLP is integrative: Linguistic interpretations depend on a large amount of non-linguistic information (e.g., world knowledge). The signs used in HPSG (typed feature structures) can incorporate both linguistic and non-linguistic information using the same formal representation.

Psycholinguistic Evidence for HPSG (cont.)

3. Human language processing is order-independent: There is no fixed sequence in which pieces of information are consulted and incorporated into linguistic interpretation. HPSG is a declarative model, so information can be added in any order.

4. HLP is reversible: We can produce and understand the same kinds of utterances. The grammar of HPSG is process-neutral. It can be used for either production or comprehension.