

# Syntactic Theory

Lecture 1 (28.10.2008)

**PD Dr.Valia Kordoni**

Email: [kordoni@coli.uni-sb.de](mailto:kordoni@coli.uni-sb.de)

<http://www.coli.uni-saarland.de/courses/syntactic-theory-08/>

# Syntax: What does it mean?

We can view syntax/syntactic theories 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 treated/manipulated

# Syntactic Analysis

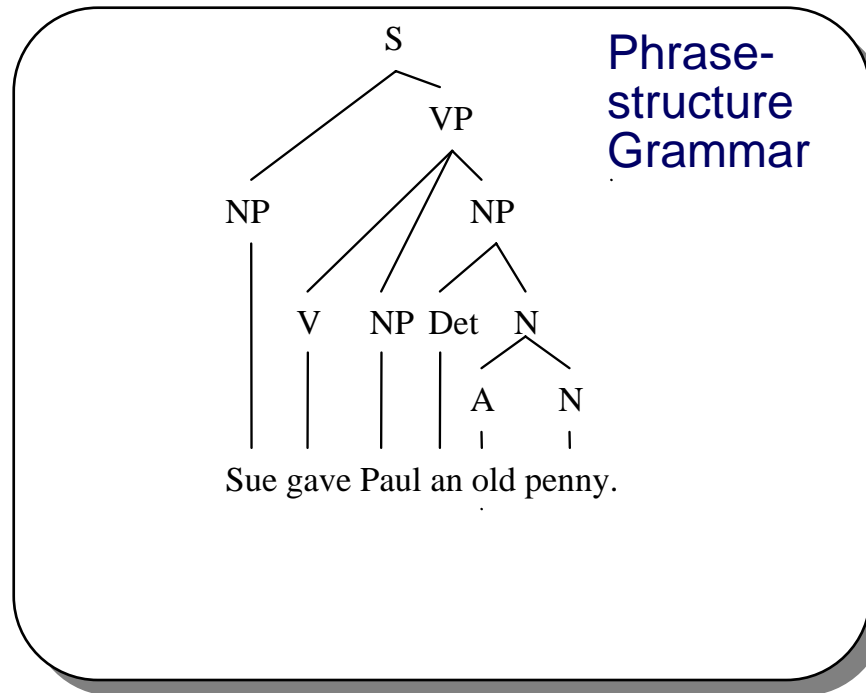
- Focus on collection of words and rules with which we generate strings of those words, i.e., sentences (generative grammar)
- Syntax attempts to capture the nature of those rules
  1. Colourless green ideas sleep furiously.
  2. \*Furiously sleep ideas green colourless.
- What generalisations are needed to capture the difference between grammatical and ungrammatical sentences?

# Phrase Structure Grammars (PSGs)

- Grouping, or constituency, is used

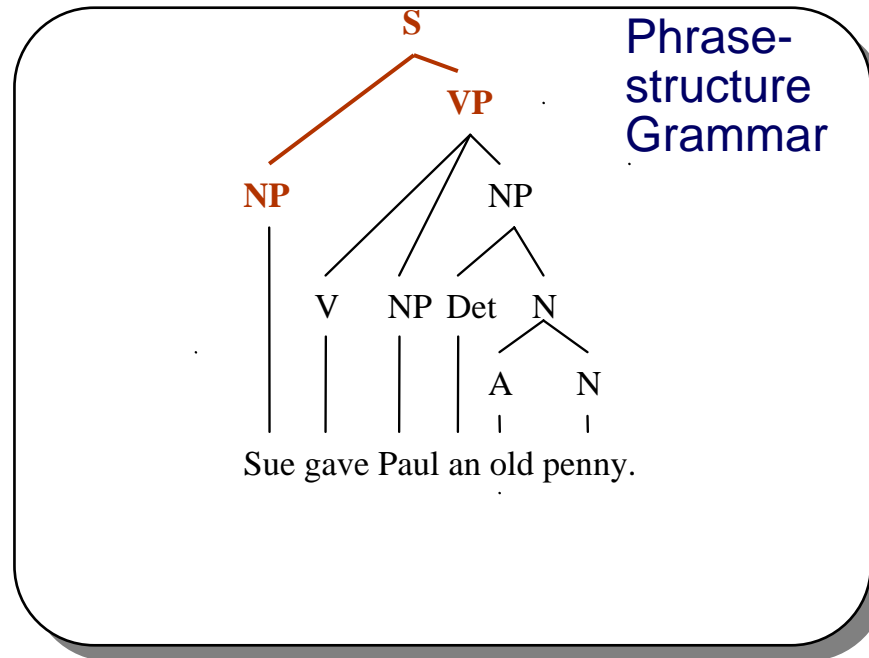
(1) Sue gave Paul an old penny.

# Phrase Structure Grammars (PSGs)



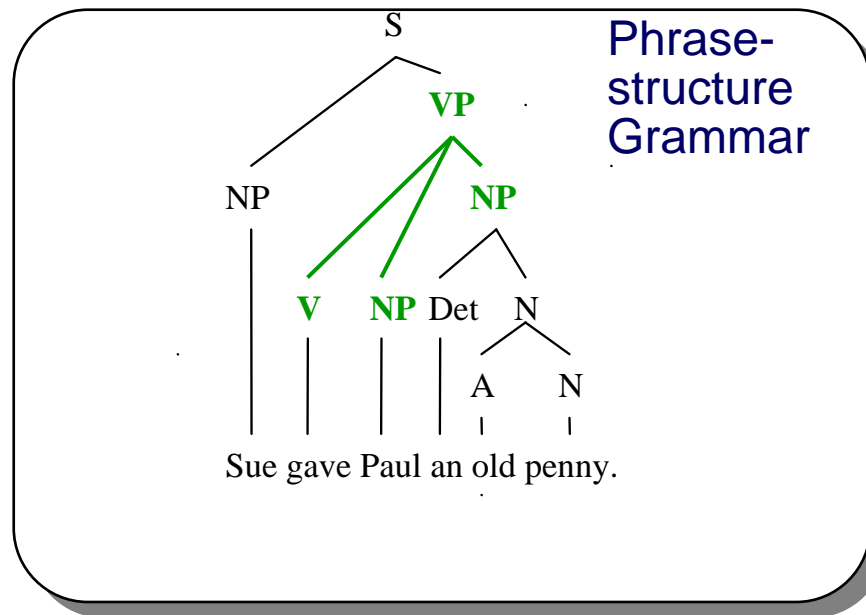
$S \rightarrow NP VP$

# Phrase Structure Grammars (PSGs)



$S \rightarrow NP VP$

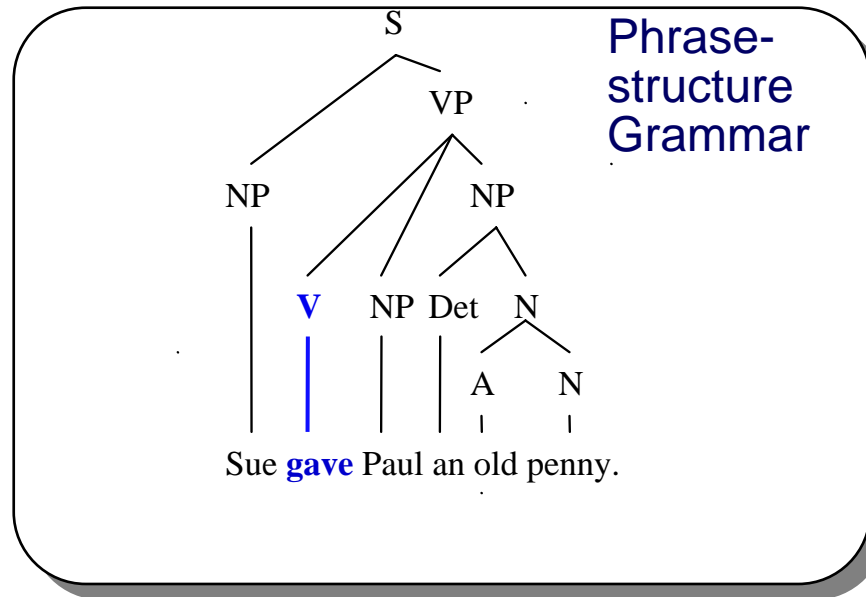
# Phrase Structure Grammars (PSGs)



$S \rightarrow NP VP$

$VP \rightarrow V NP NP$

# Phrase Structure Grammars (PSGs)



$S \rightarrow NP VP$

$VP \rightarrow V NP NP$

$V \rightarrow \text{gave}$

# The Transformational Tradition

Roughly speaking, **transformational syntax** (GB = Government and Binding, P&P = Principles and Parameters,...) has focused on the following:

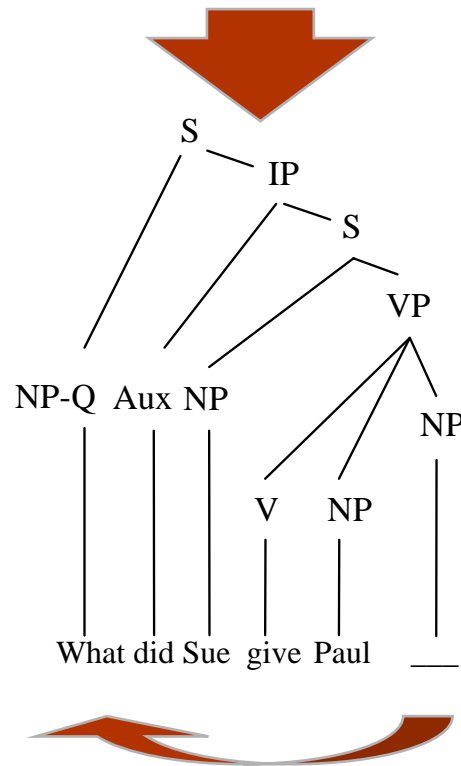
- Explanatory adequacy: the data must fit with a deeper model, that of universal grammar
- Psychological: does the grammar make sense in light of what we know of how the mind works?
- Theory-driven: data should ideally fit with a theory already in place (often based on English)

# 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

# The Transformational Tradition (cont.)

Sue gave Paul an old penny



# The Transformational Tradition (cont.)

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

# Making it computational

How is a syntactic 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 Machine Translation (MT) and 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

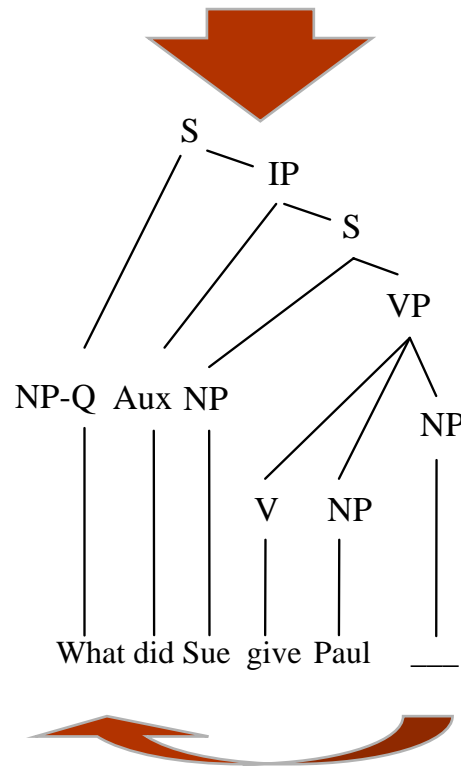
Only have one (surface) syntactic level

- 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:

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- ...

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 Categorical 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 Categorical Grammar (CCG)

- Categorical 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)