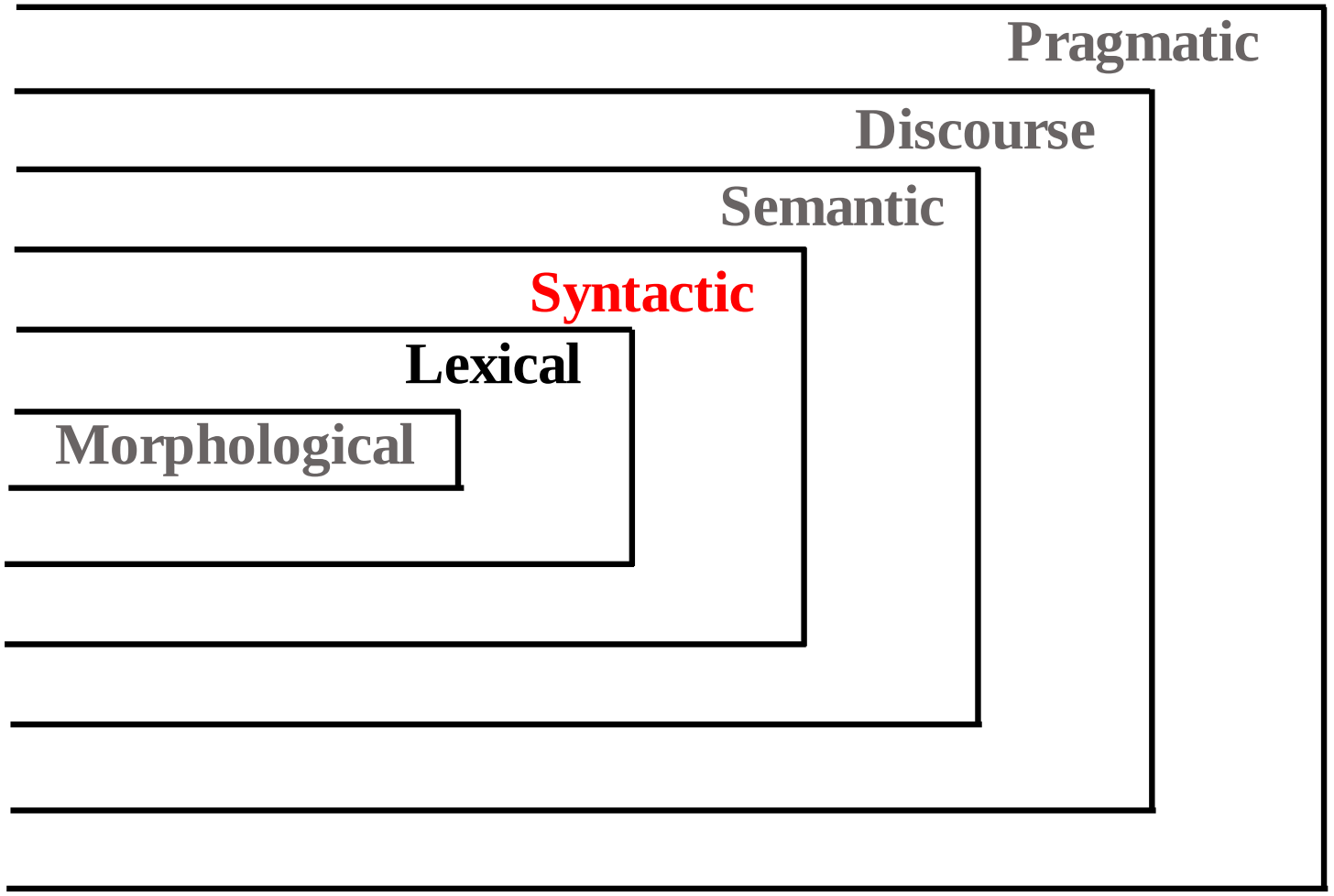
CONTEXT FREE GRAMMARS

LU XIAO
LXIA004@SYR.EDU
213 HINDS HALL

ADOPTED SOME MATERIALS DEVELOPED IN PREVIOUS COURSES BY NANCY MCCRACKEN, LIZ LIDDY AND OTHERS; AND SOME INSTRUCTOR RESOURCES FOR THE BOOK "SPEECH AND LANGUAGE PROCESSING" BY DANIEL JURAFSKY AND JAMES H. MARTIN

SYNCHRONIC MODEL OF LANGUAGE



Syntactic Analysis

- Syntax expresses the way in which words are arranged together

- Use grammars/rules that embody generalizations that hold for the symbols and combinations of symbols in a language for constructing acceptable sentences
  - grammar is most closely identified with syntax, but may contain elements of all levels of language

- By grammar, or syntax, we mean the kind of implicit knowledge of your native language that you had mastered by the time you were 3 years old without explicit instruction
  - Not the kind of stuff you were later taught in "grammar" school
    - *I saw you yesterday* vs. *you yesterday I saw*
  - Chomsky: syntactic structure can be independent on the meaning of the sentence
    - *Colorless green ideas sleep furiously* (grammatically correct) *vs. Furiously sleep ideas green colorless* (grammatically incorrect)

Why Should You Care?

- Grammars (and parsing) are key components in many applications:
  - Grammar checkers
  - Dialogue management
  - Question answering
  - Information extraction
  - Machine translation

Constituency

- The basic idea: groups of words within utterances can be shown to act as single units.
- And in a given language, these units form coherent classes that can be shown to behave in similar ways

For example, it makes sense to the say that the following are all noun phrases in English...

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

Why? One piece of evidence is that they can all precede verbs.

This is external evidence

Context-free Grammars (CFGs)

- Also known as Phrase structure grammars or Backus-Naur form
- Capture **constituency and ordering**
  - Constituency is How do words group into units and what we say about how the various kinds of units behave
    - A constituent is a sequence of words that behave as a unit
      - John talked [to the children] [about drugs].
      - John talked [about drugs] [to the children].
      - *John talked drugs to the children about (random reorder)
    - Constituents can be expanded or substituted for:
      - I sat [on the box/right on top of the box/there]
  - Ordering is What are the rules that govern the ordering of words and bigger units in the language

Notations of Context-free Grammar

- Non-terminal symbols

  S, NP, VP, etc. representing the constituents or categories of phrases

- Terminal symbols

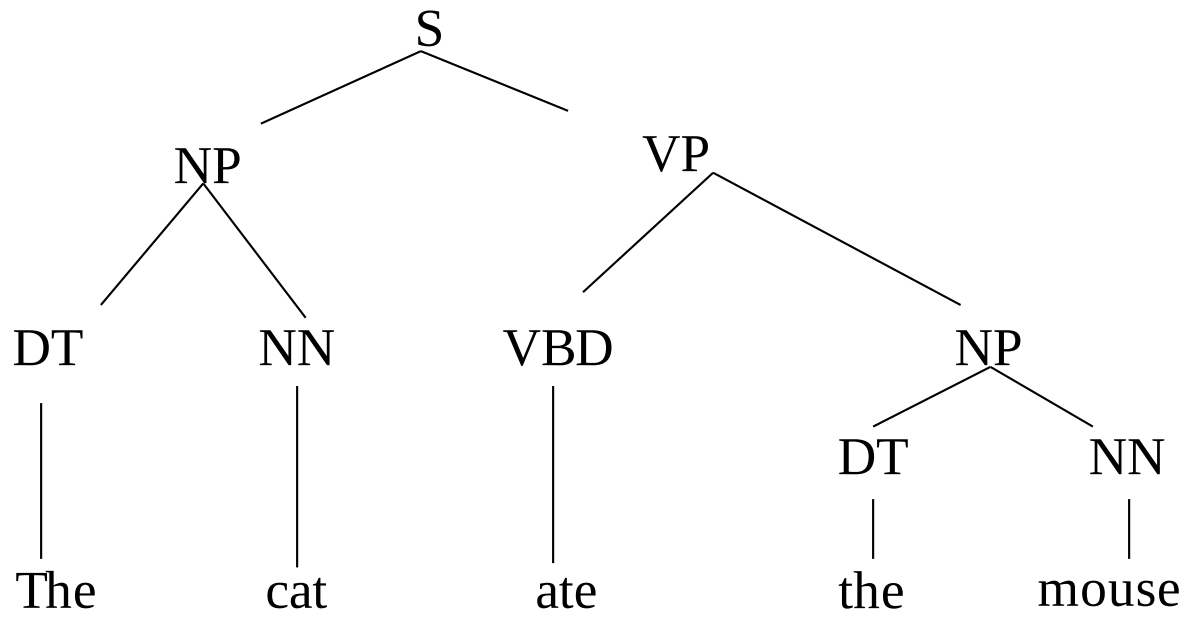
  car, man, house, representing words in the lexicon

- Rewrite rules / productions

  S → NP VP | VP

  The rewrite rules will include lexical insertion rules (e.g. N -> *car* | *man* | *house*)

- A designated start symbol S



The phase structure rules underlying this analysis are as follows:

S ⟶ NP VP

NP ⟶ DT NN

VP ⟶ VBD NP

DT = The

NN = cat

NN = mouse

Verb = ate

Parsing a sentence using simple phrase structure rules

Key Constituents For English

- English has headed phrase structure
  - X-bar theory: in natural languages, phrases are headed by particular kinds of words with modifiers and qualifiers around them (specifiers, adjuncts, and complements)
- Noun Phrases NP → … NN* …
- Verb Phrases VP → … VB* …
- Adjective Phrases ADJP → … JJ* …
- Adverb Phrases ADVP → … RB* …
- Sentences (and clauses): SBAR → S | SINV | SQ …
  - Sentences, inverted sentences, direct questions, … can also appear in larger clause structure SBAR where sentence is preceded by *that*
- Plus minor phrase types:
  - QP (quantifier phrase in NP), PP (prepositional phrase), CONJP (multi word constructions: *as well as*), INTJ (interjections), etc.

e.g. Penn Treebank Constituent Tags:
http://www.surdeanu.info/mihai/teaching/ista555-fall13/readings/PennTreebankConstituents.html

Noun Phrases

- Noun phrases have a head noun with pre and post-modifiers
  - Determiners, Cardinals, Ordinals, Quantifiers and Adjective Phrases are all optional, indicated here with parentheses

    NP -> (DT) (Card) (Ord) (Quan) (AP) Noun

    Noun -> NN | NNP | NNPS | NNS *(the four noun POS tags)*
  - Post-modifiers include prepositional phrases, gerundive phrases, and relative clauses

    the man [from Moscow]

    any flights [arriving after 11pm] (gerundive)

    the spy[who came in from the cold] (relative clause)

Some examples on these slides are from the Jurafsky and Martin text and from Jim Martin's online course materials.

Recursive Rules

- One type of Noun phrase is a Noun Phrase followed by a Prepositional phrase
  - * NP -> NP PP
    PP -> Prep NP
- Of course, this is what makes syntax interesting

  flights from Denver

  flights from Denver to Miami

  flights from Denver to Miami in February

  flights from Denver to Miami in February on a Friday

  flights from Denver to Miami in February on a Friday under $300

  flights from Denver to Miami in February on a Friday under $300 with lunch

  - Syntax trees for these examples also need rules for NP -> Noun, etc.

* This grammar illustrates the recursion, but may not give the best derivation for these phrases!

Verb Phrases

- Simple Verb phrases

VP -> Verb	*leave*
| Verb NP	*leave Boston*
| Verb NP PP	*leave Boston in the morning*
| Verb PP	*leave in the morning*

- Verbs may also be followed by a clause

  VP -> Verb S

  I think I would like to take a 9:30 flight

- The phrase or clause following a verb is sometimes called the complementizer

Sentences

- Sentences
  - Declaratives: A plane left

    S -> NP VP
  - Imperatives: Leave!

    S -> VP
  - Yes-No Questions: Did the plane leave?

    S -> Aux NP VP
  - WH Questions: When did the plane leave?

    S -> WH Aux NP VP

Conjunctive Constructions

- S -> S and S
  - John went to NY and Mary followed him
- NP -> NP and NP
- VP -> VP and VP
- …
- In fact the right rule for English is
  X -> X and X

Definition

- More formally, a CFG consists of

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol**

Why Context-Free?

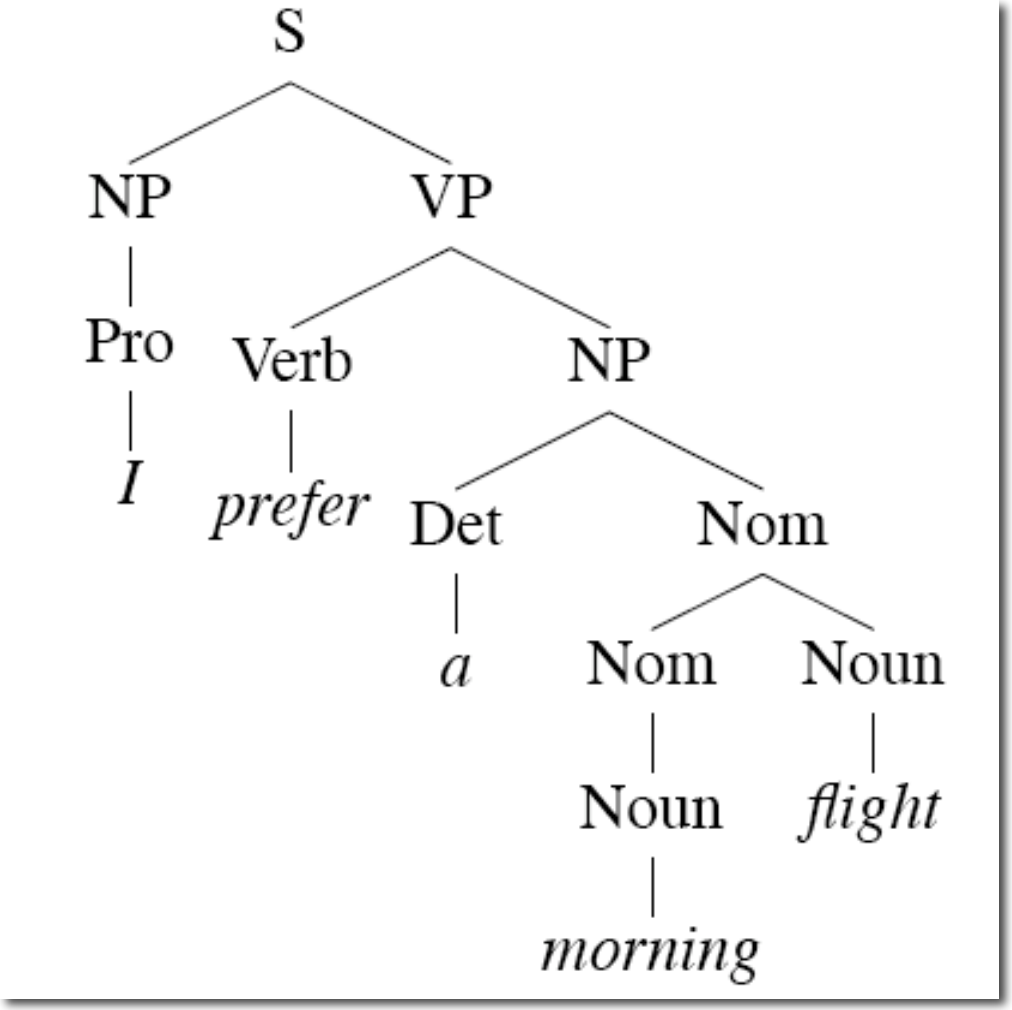
The non-terminal on the left-hand side of a rule can be replaced by the right-hand side regardless of context

Context-sensitive grammars allow context to be placed on the left-hand side of the rewrite rule

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Derivations

- A derivation is a sequence of rules applied to a string that *accounts* for that string
  - Covers all the elements in the string
  - Covers only the elements in the string



Derivation Of Syntax From Grammar Rules

- *Given the grammar and a sentence,*
  Show top-down derivation of a parse tree.

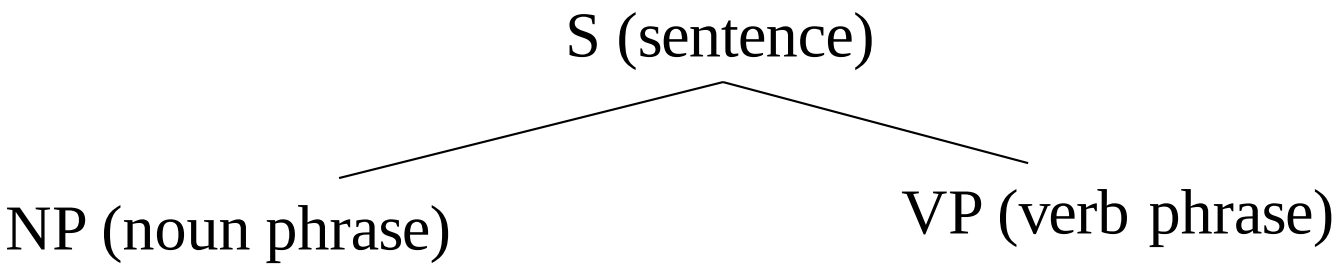
the *man* *eats* *the* *apple*

Context Free Grammar Rules (for this example):

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Top Down Derivation – Starts With S



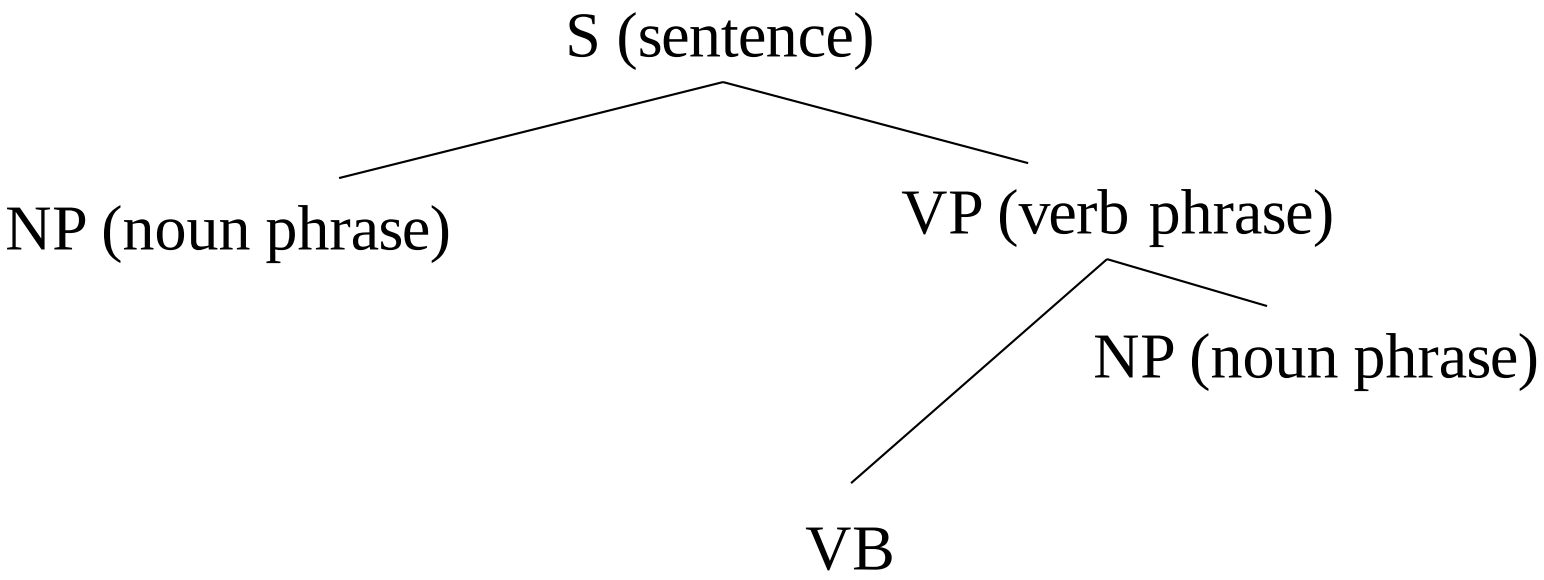
the *man* *eats* *the* *apple*

Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Top Down Derivation – Add Rule For VP



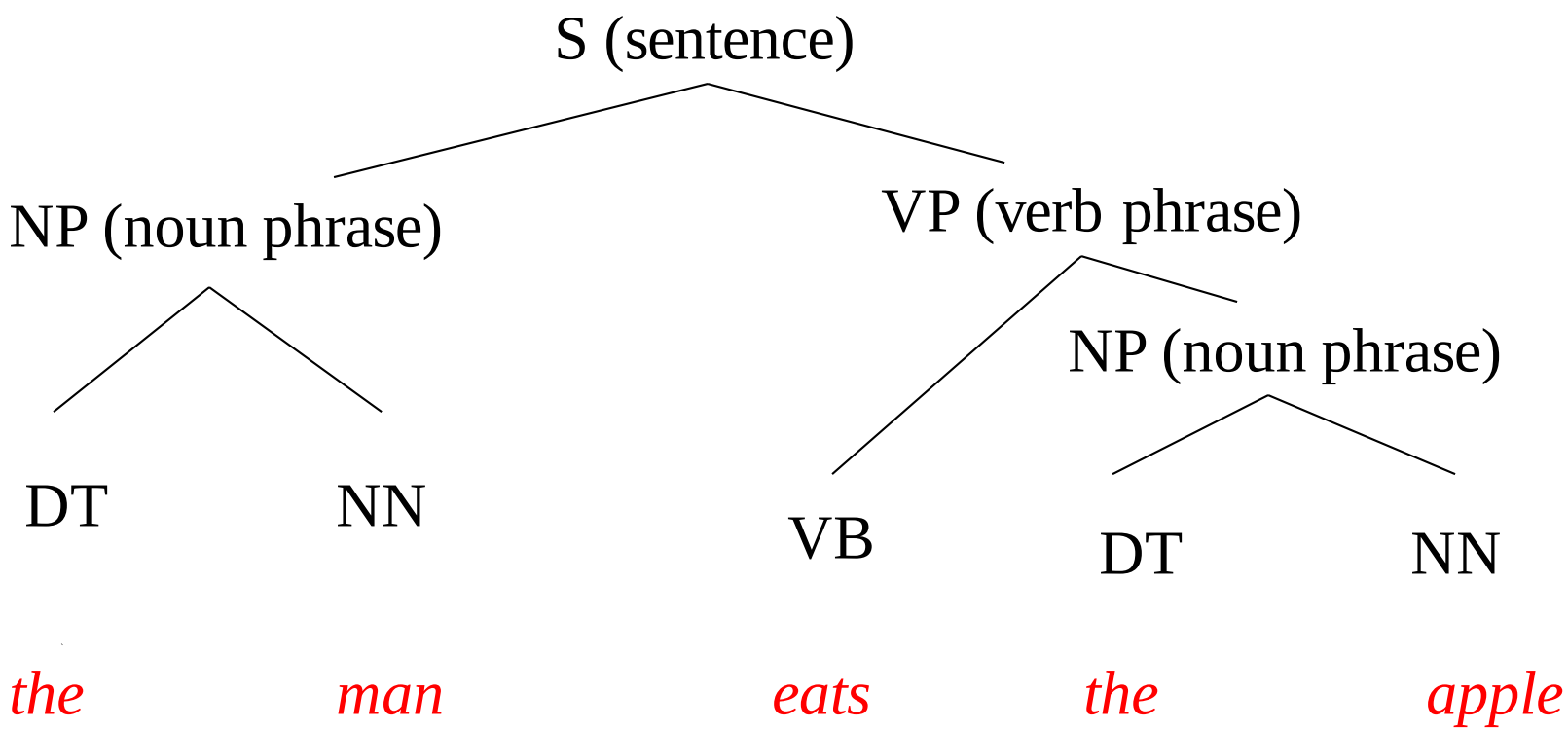
the *man* *eats* *the* *apple*

Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Top Down Derivation – Add Rules For NP

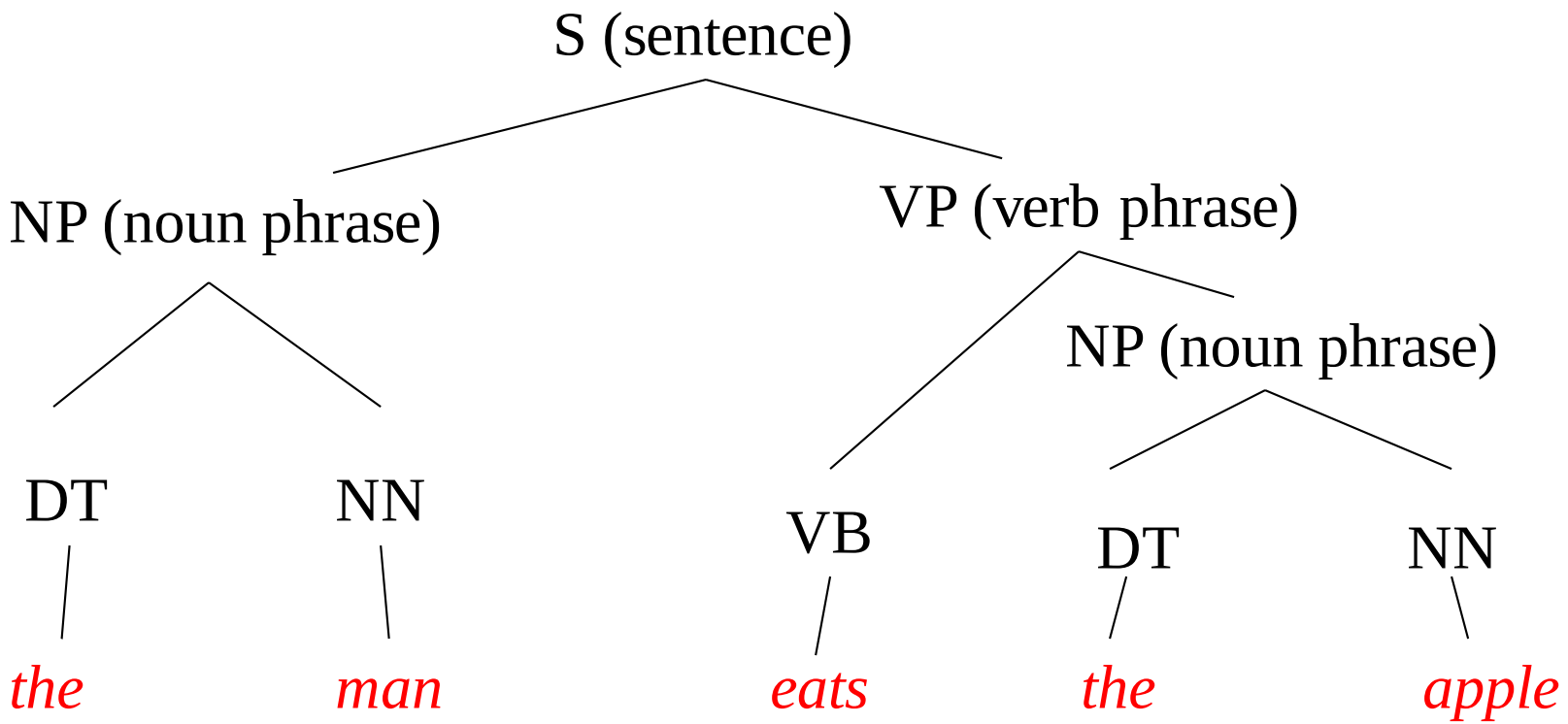


Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Top Down Derivation – Add POS/Lexical Rules



Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Derivation Of Syntax From Grammar Rules

- *Given the grammar and a sentence,*
  Show bottom-up derivation of a parse tree.

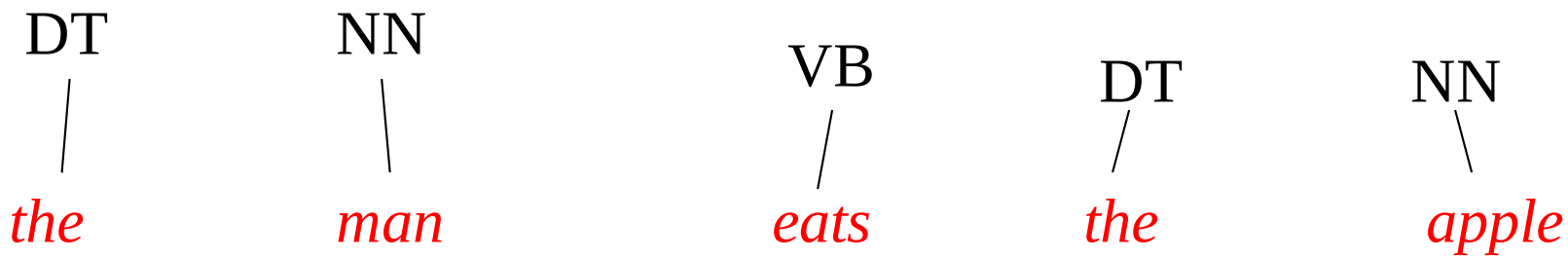
the *man* *eats* *the* *apple*

Context Free Grammar Rules (for this example):

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Bottom Up Derivation – Start With POS/Lexical Rules

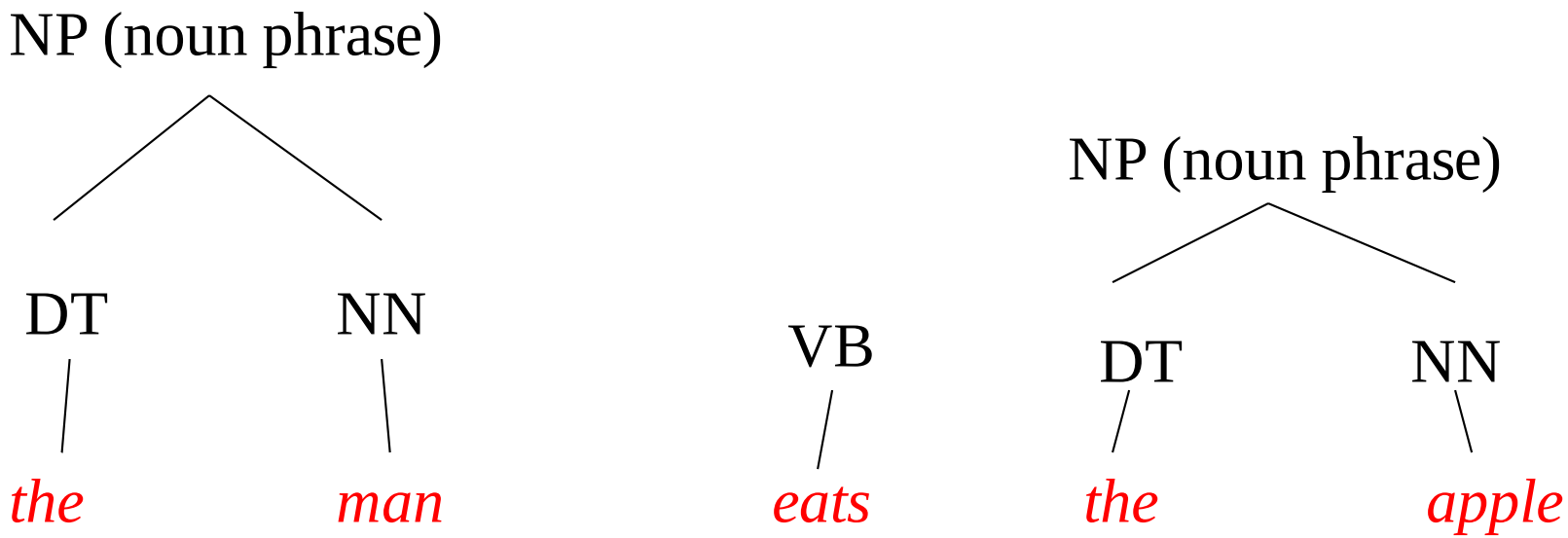


Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Bottom Up Derivation – Add NP Rules

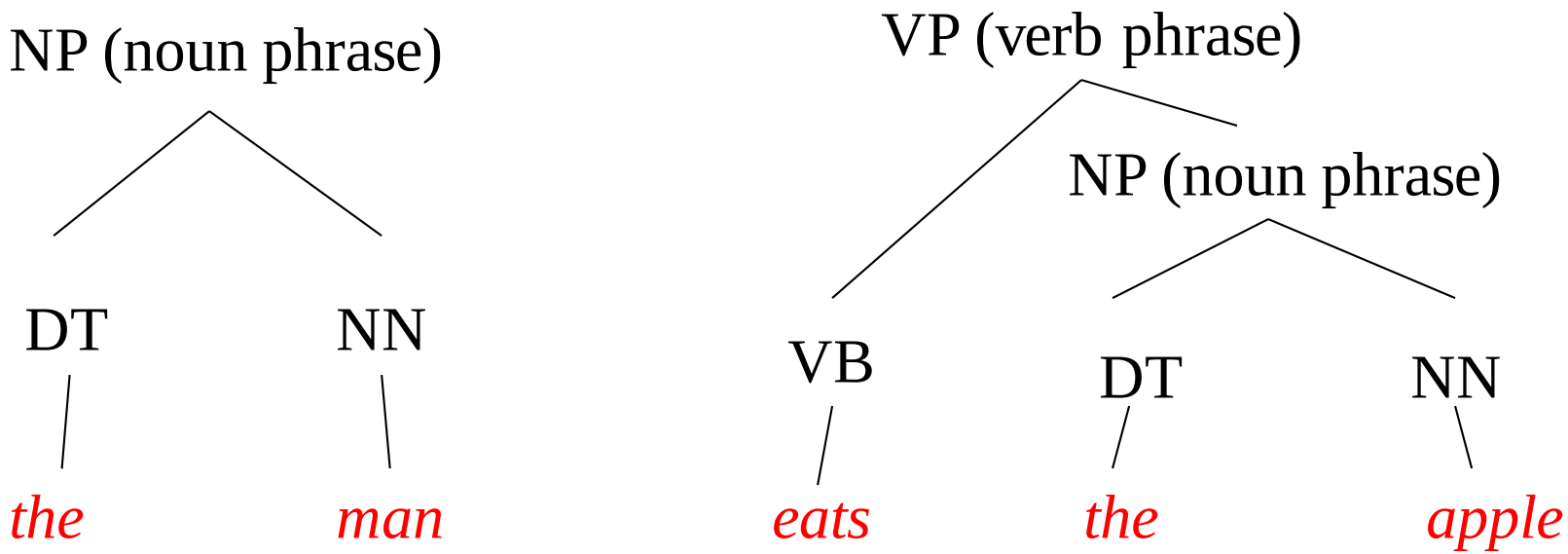


Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Bottom Up Derivation – Add VP Rule

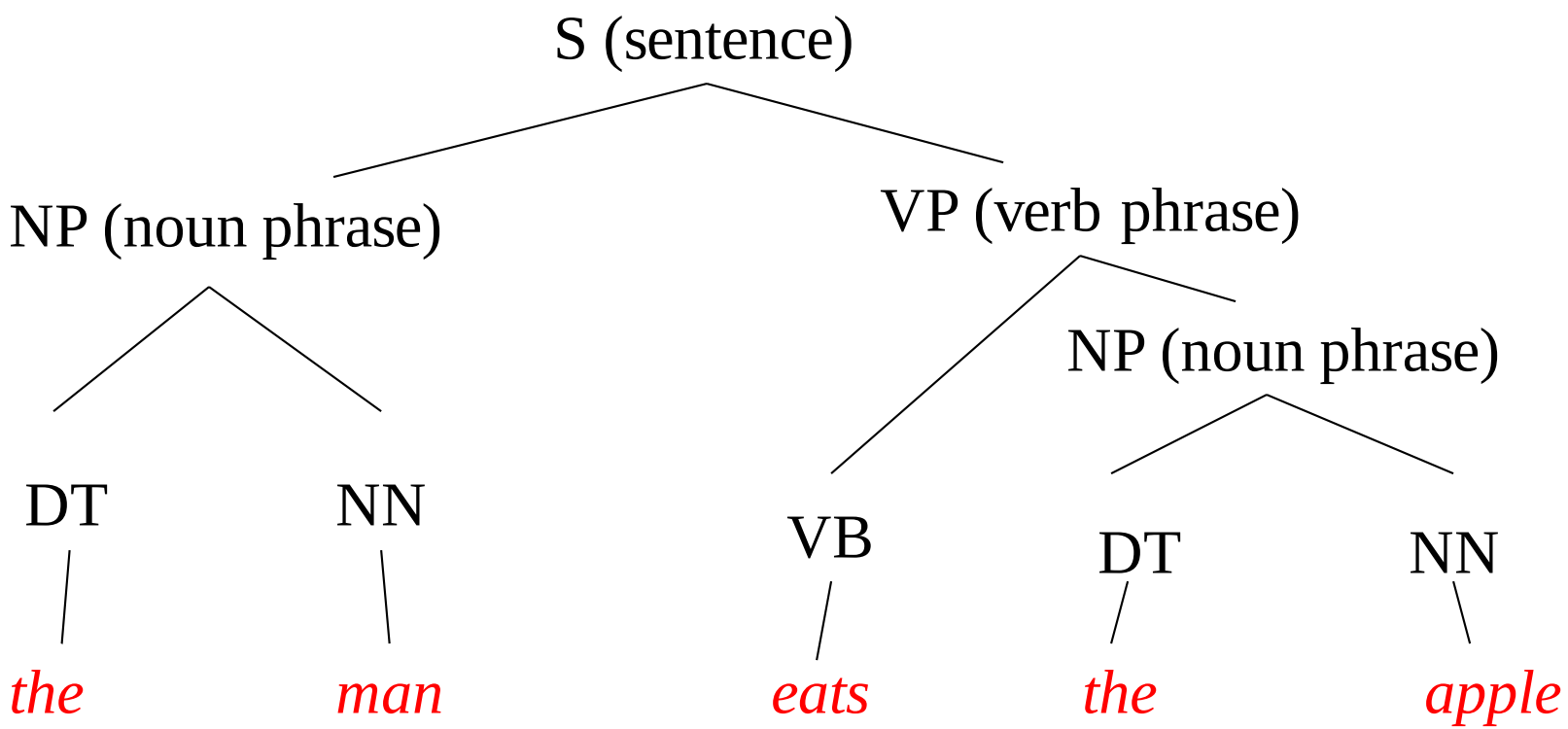


Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Bottom Up Derivation – Add S Rule



Context Free Grammar Rules:

S → NP VP
NP → DT NN
VP → VB NP
VP → VB

DT → *the* | …
NN → *man* | apple | … (add words)
VB → *eats* | …

Notations For (Constituent) Syntactic Structure

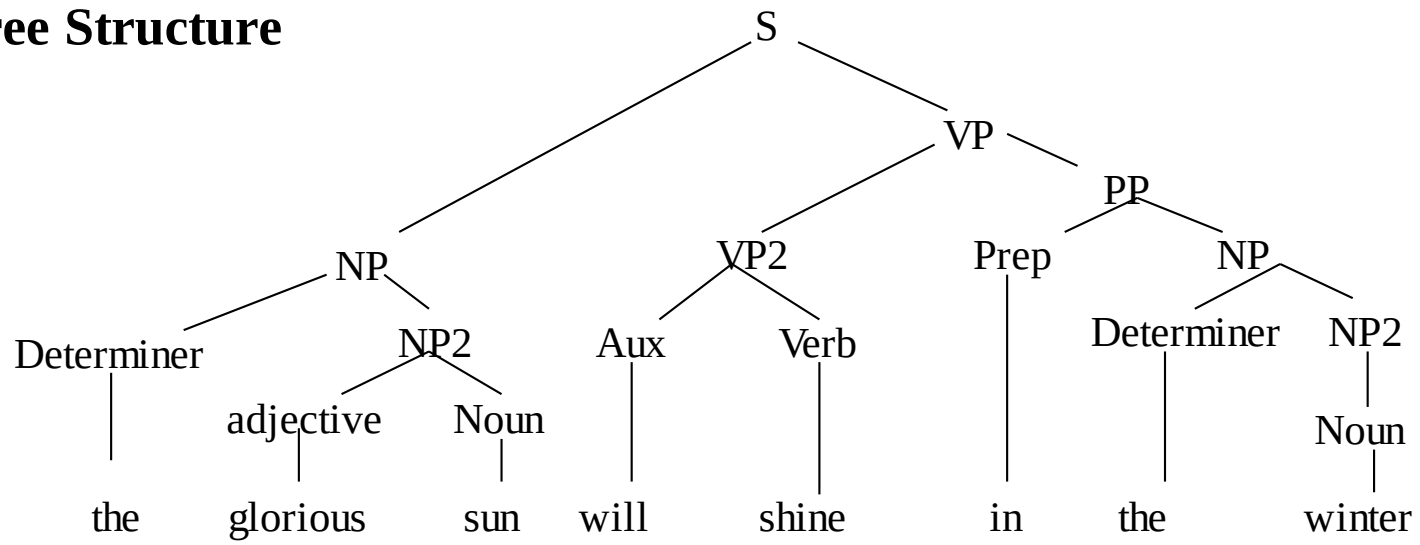
Bracketed text

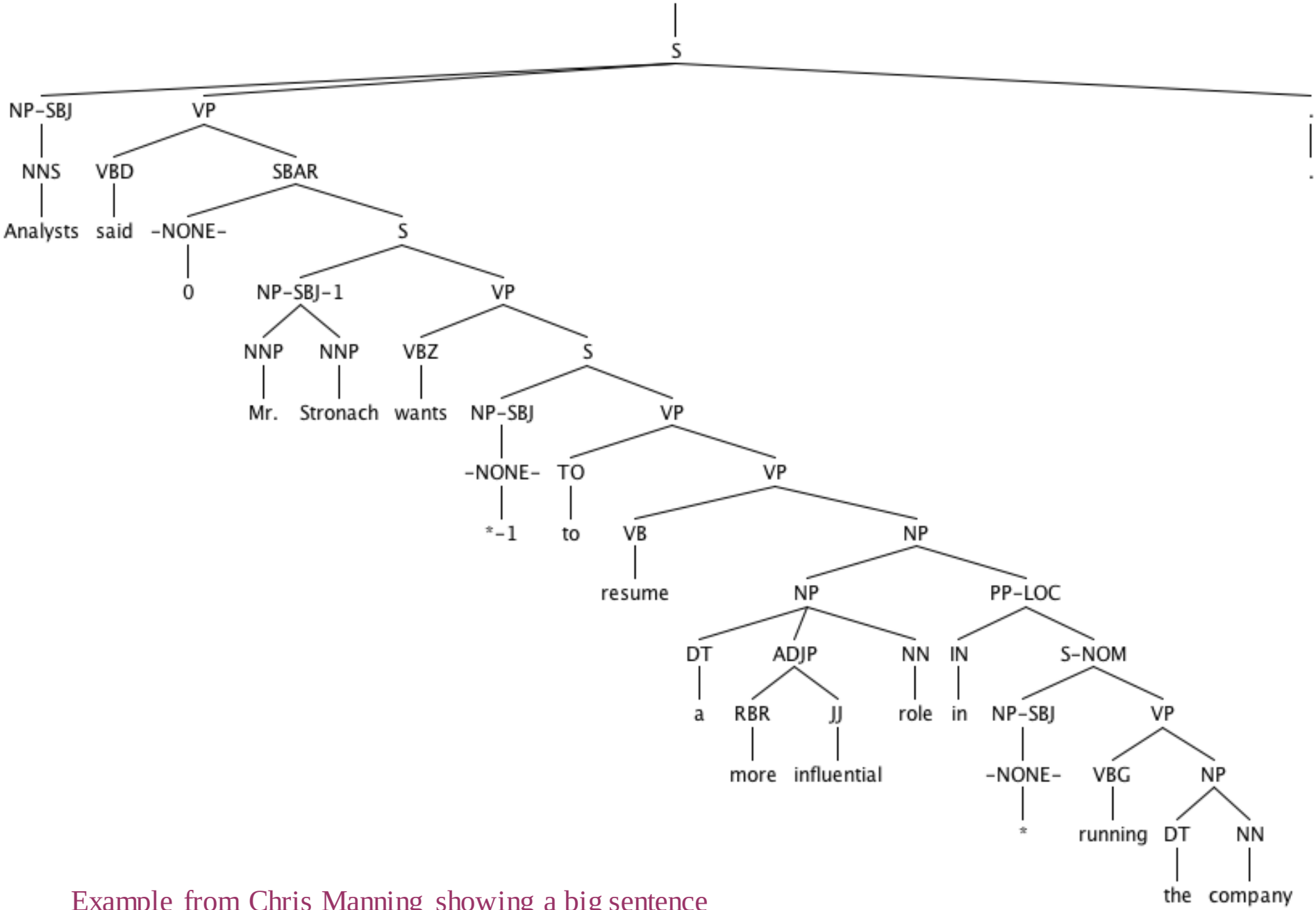
[$_{S}$ [$_{NP}$ the [$_{NP2}$ glorious sun]] [$_{VP}$ [$_{VP2}$ will shine] [$_{PP}$ in [$_{NP}$ the [$_{NP2}$ winter]]]]]

Indented bracketed text

```
(S
  (NP (DT The) (JJ glorious) (NN sun))
  (VP (MD will)
    (VP (VB shine)
      (PP (IN in)
        (NP (DT the) (NN winter)))))))
```

Tree Structure



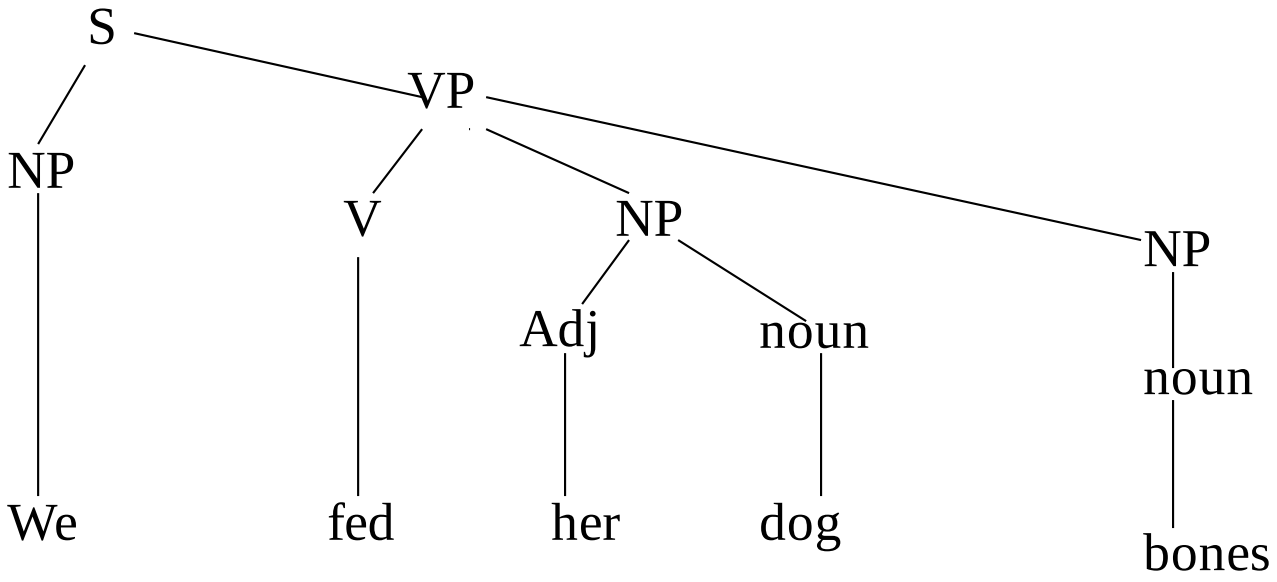


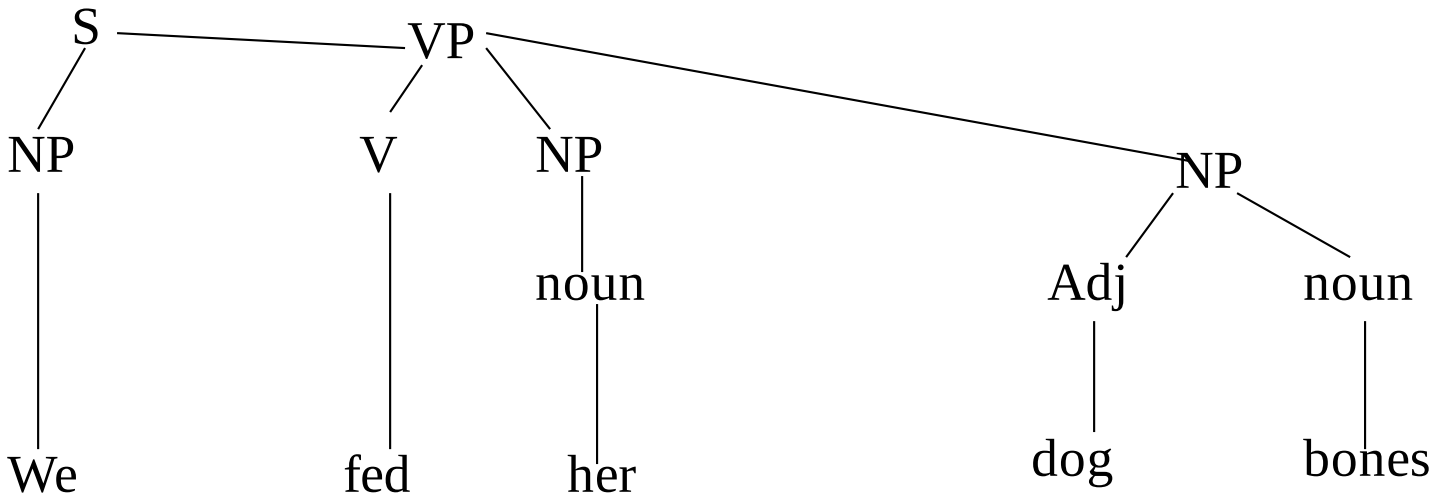
Example from Chris Manning showing a big sentence with nested constituents and empty elements.

Generativity Vs. Parsing

- You can view these rules as either synthesis or analysis machines
  - Generate strings in the language
  - Reject strings not in the language
  - Impose structures (trees) on strings in the language
- The latter two are the analysis tasks of parsing
  - Parsing is the process of finding a derivation (i. e. sequence of productions) leading from the START symbol to a TERMINAL symbol (or TERMINALS to START symbol)
  - Shows how a particular sentence ***could be*** generated by the rules of the grammar
  - If sentence is structurally ambiguous, more than one possible derivation is produced

Syntactic Ambiguity: We fed her dog bones





Problems

- Context-Free Grammars can represent many parts of natural language adequately
- Here are some of the problems that are difficult to represent in a CFG:
  - Agreement
  - Subcategorization
  - Movement

Agreement

- This dog
- Those dogs
- *This dogs
- *Those dog

- This dog eats
- Those dogs eat
- *This dog eat
- *Those dogs eats

- In English,
  - subjects and verbs have to agree in person and number
  - Determiners and nouns have to agree in number
- Many languages have agreement systems that are far more complex than this.
- Solution can be either to add rules with agreement or to have a layer on the grammar called the features

Subcategorization

- Subcategorization expresses the constraints that a particular verb (sometimes called the predicate) places on the number and syntactic types of arguments it wants to take (occur with).
  - Sneeze: John sneezed
  - Find: Please find [a flight to NY]$_{NP}$
  - Give: Give [me]$_{NP}$[a cheaper fare]$_{NP}$
  - Help: Can you help [me]$_{NP}$[with a flight]$_{PP}$
  - Prefer: I prefer [to leave earlier]$_{TO\text{-}VP}$
  - Told: I was told [United has a flight]$_{S}$
  - …

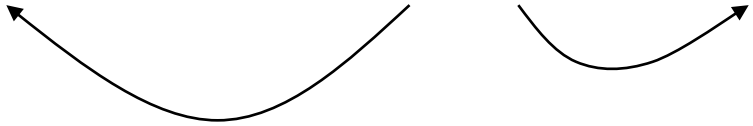
Subcategorization

- The various rules for VPs *overgenerate*.
  - They permit the presence of strings containing verbs and arguments that don't go together
  - For example VP -> V NP therefore

    Sneezed the book is a VP since "sneeze" is a verb and "the book" is a valid NP – sneeze is not a verb that can be followed by a noun phrase
- Now *overgeneration* is a problem for a generative approach.
  - The grammar should represent all and only the strings in a language
- From a practical point of view… Not so clear that there's a problem - generally people produce the sentences that have proper considerations

Movement

- Consider the verb "booked" in the following example:
  - [[My travel agent]$_{NP}$ [booked [the flight]$_{NP}$]$_{VP}$]$_{S}$



- i.e. "book" is a straightforward transitive verb. It expects a single NP arg within the VP as an argument, and a single NP arg as the subject.

Movement

- But what about?
  - Which flight do you want me to have the travel agent book?
- The direct object argument to "book" isn't appearing in the right place. It is in fact a long way from where it's supposed to appear.
- And note that it's separated from its verb by 2 other verbs.

Stanford Parser output
http://nlp.stanford.edu:8080/parser/

```
Which/WDT flight/NN do/VBP you/PRP want/VB me/PRP to/TO
have/VB the/DT travel/NN agent/NN book/NN ?/.
```

The Point About CFGs

- CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English.
- But there are problems
  - that can be dealt with adequately, although not elegantly, by staying within the CFG framework.
- There are simpler, more elegant, solutions that take us out of the CFG framework (beyond its formal power)

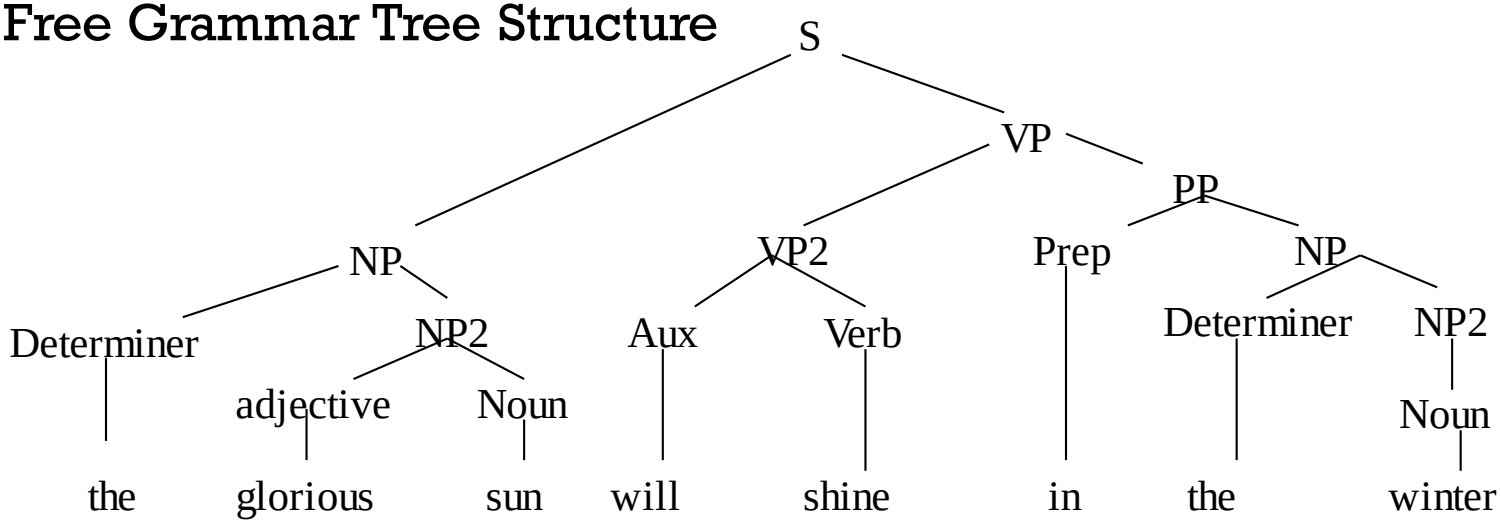
Dependency Grammars, An Alternative To CFGs

Dependency Grammars

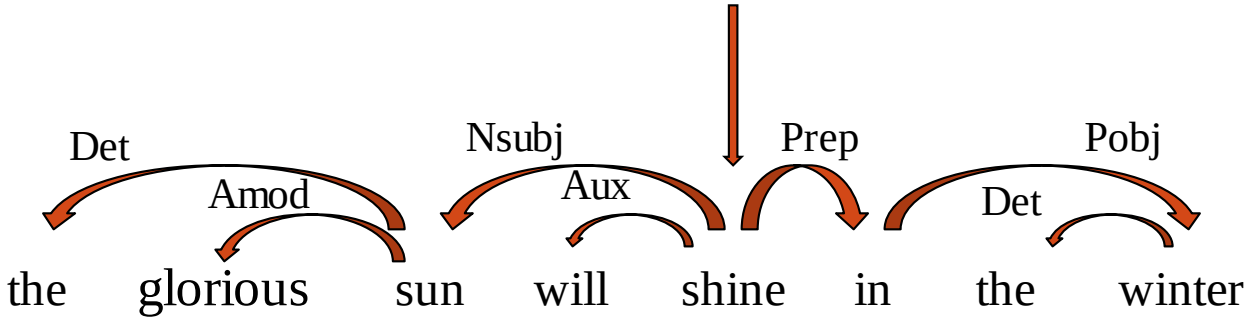
- Dependency grammars offer a different way to represent syntactic structure
  - CFGs represent constituents in a parse tree that can derive the words of a sentence
  - Dependency grammars represent syntactic dependency relations between words that show the syntactic structure
  - Typed dependency grammars label those relations as to what the syntactic structure is
- Syntactic structure is the set of relations between a word (aka the head word) and its dependents.

Examples

- Context Free Grammar Tree Structure



- Dependency Relation Structure



Note that the head word of a sentence is the verb.

Dependency Relations

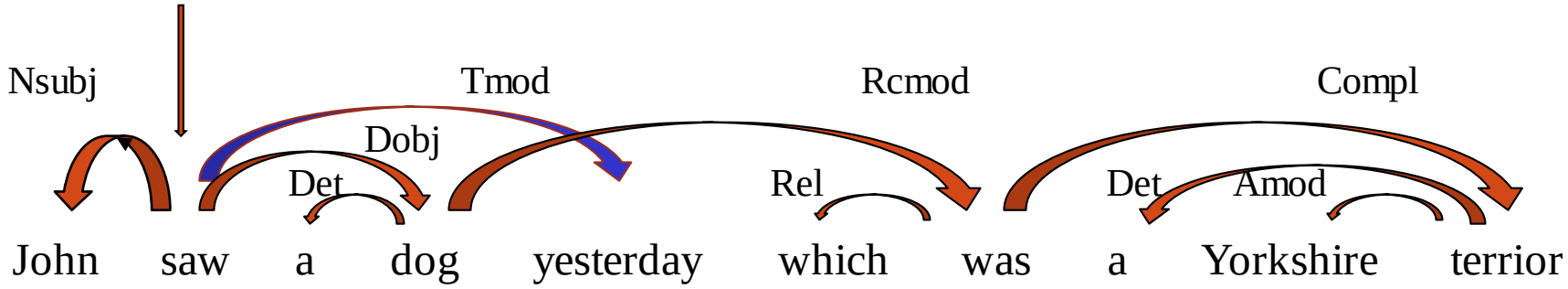
- The set of Grammar Relations has varied in number
  - 48 in the Stanford dependency parser
  - 59 in Minipar, a dependency parser from Dekang Lin
  - 106 in Link, a related link grammar parser from CMU
- The examples on the previous page used those from the Stanford dependency parser
  - De Marneffe, MacCartney and Manning, Generating Typed Dependency Parses from Phrase Structure Parses, LREC (Language Resources and Evaluation Conference), 2006.

Dependency Relations Examples:

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Projective Vs. Non-projective

- In the dependency graph as depicted in the previous example, with the words in sentence order, if no arcs cross, then it is a projective tree
- If there are crossing arcs, then it is a non-projective tree



- Non-projective trees are a problem for parsing, not for expressive power of the grammar
- CoNLL (Conference on Natural Language Learning) 2006 had dependency parsing as the shared task on 13 languages, not including English. Out of the languages which had non-projective sentences in the treebanks, from 0.5% to 5% of the sentences were non-projective

Dependency Grammar Vs. CFG

- Dependency grammars and CFGs are strongly equivalent
  - Generate the same sentences and make the same structural analysis
    - Haim Gaifman, 1965, "Dependency systems and phrase structure systems".
- Equivalent provided that the CFGs are restricted in that one word or phrase can be designated as its "head"
  - This restriction also accepted by linguists in X-bar theory
    - Proposed by Chomsky and further developed by Ray Jackendoff, 1977, "X-bar-Syntax: A Study of Phrase Structure"
  - Note that the head of a noun phrase is a noun, the head of a verb phrase is a verb, etc.