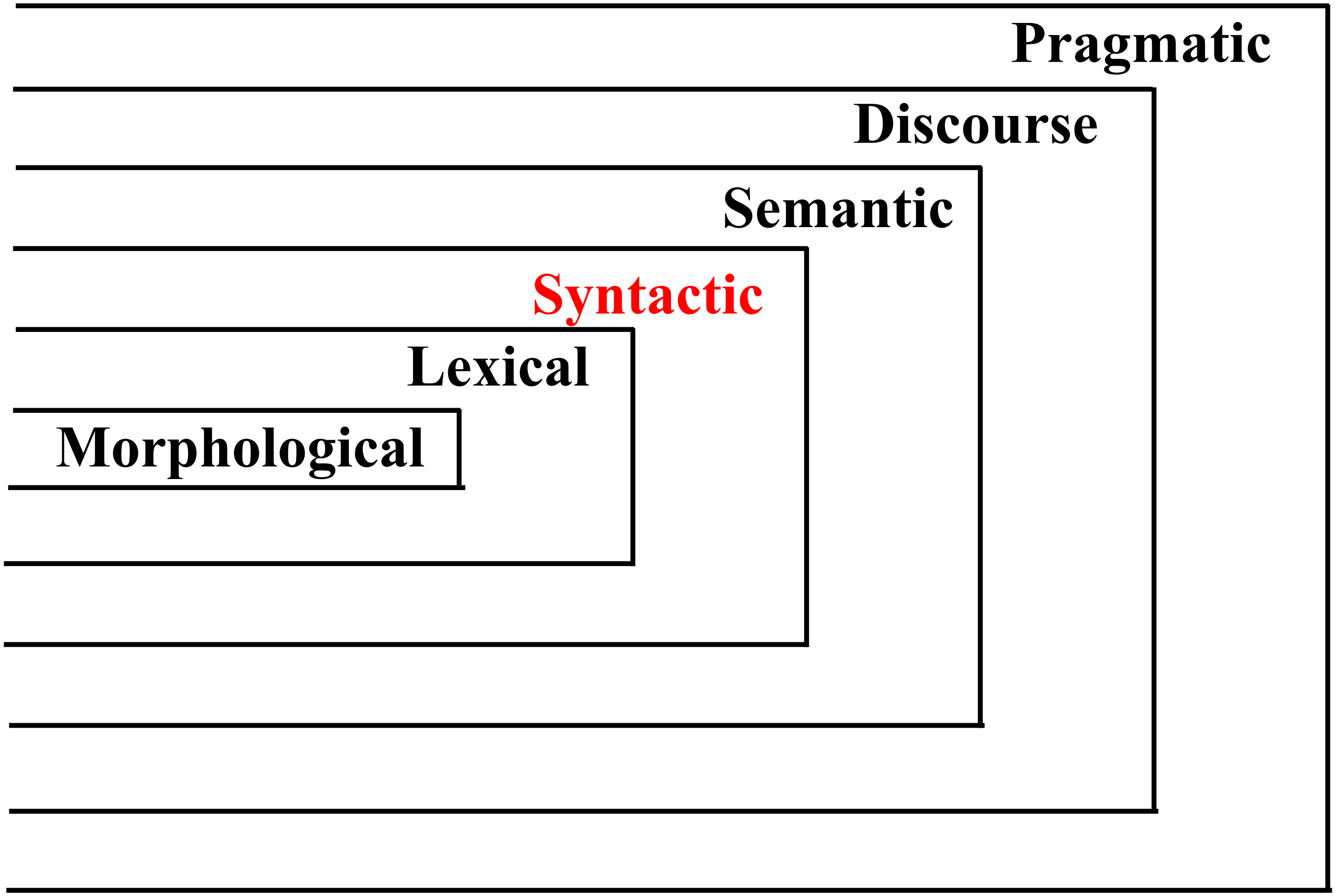
Introduction to
Context Free Grammars

Synchronic Model of Language



Syntactic Analysis

- Syntax expresses the way in which words are arranged together.
- The kind of implicit knowledge of your native language that you had mastered by the time you were 3 or 4 years old without explicit instruction
  - Do these word sequences fit together?

    I saw you yesterday
    you yesterday I year

    colorless green ideas sleep furiously (Chomsky)
    furiously sleep ideas green colorless
- NLP uses syntax to produce a structural analysis of the input sentence

Formal Grammar

- 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
- Theoretical linguists use grammar
  - To indicate which are the well-formed sentences, defining that language
  - To show how variations of a ‘deep-structure’ are derived through ‘transformations’ on this ‘deep-structure’
  - Competence based – an ideal speaker’s internalized ability to create and understand all sentences
- Applied uses
  - To assign a structural description to the linguistic elements of which an utterance is comprised
  - “…typically not consciously modeled after any particular linguistic theory, but as descriptions of phenomena that appeared in input text.”
  - ‘Performance’ based – person’s actual use of language

Context-Free Grammars

- Capture constituency and ordering
  - Ordering is
    - What are the rules that govern the ordering of words and bigger units in the language
  - 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]
  - Other properties: Coordination, regular internal structure, no intrusion, fragments, semantics, …

Context-Free Grammar consists of:

- Non-terminal symbols

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

  car, man, house, representing words in the lexicon
  - The rewrite rules will include lexical insertion rules
    (e.g. N = *car* | *man* | *house*)
- Rewrite rules / productions

  S → NP VP | VP

  (note use of | symbol to give alternate rhs of rules)
- A designated start symbol S

- A derivation is a sequence of rewrite rules applied to a string that exactly covers the items in that string

Derivation of syntax from grammar rules

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

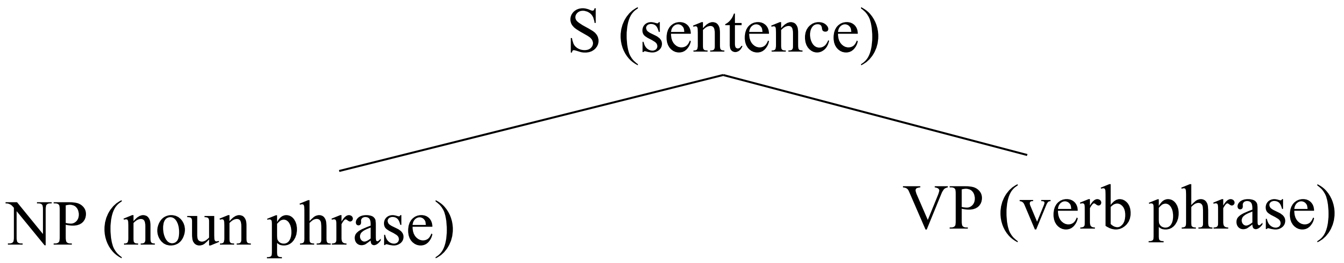
the *man* *eats* *the* *apple*

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



S (sentence)

NP (noun phrase)

VP (verb phrase)

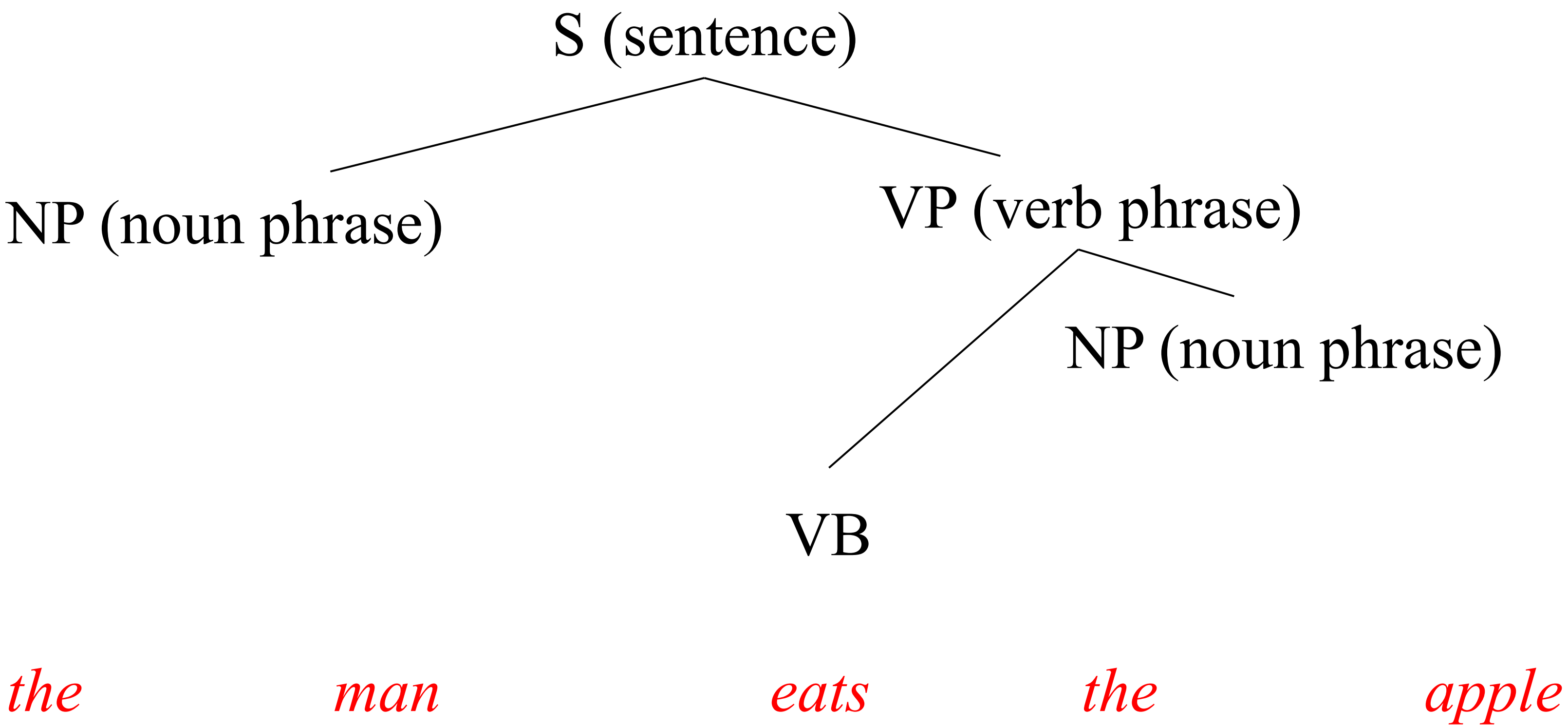
the *man* *eats* *the* *apple*

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

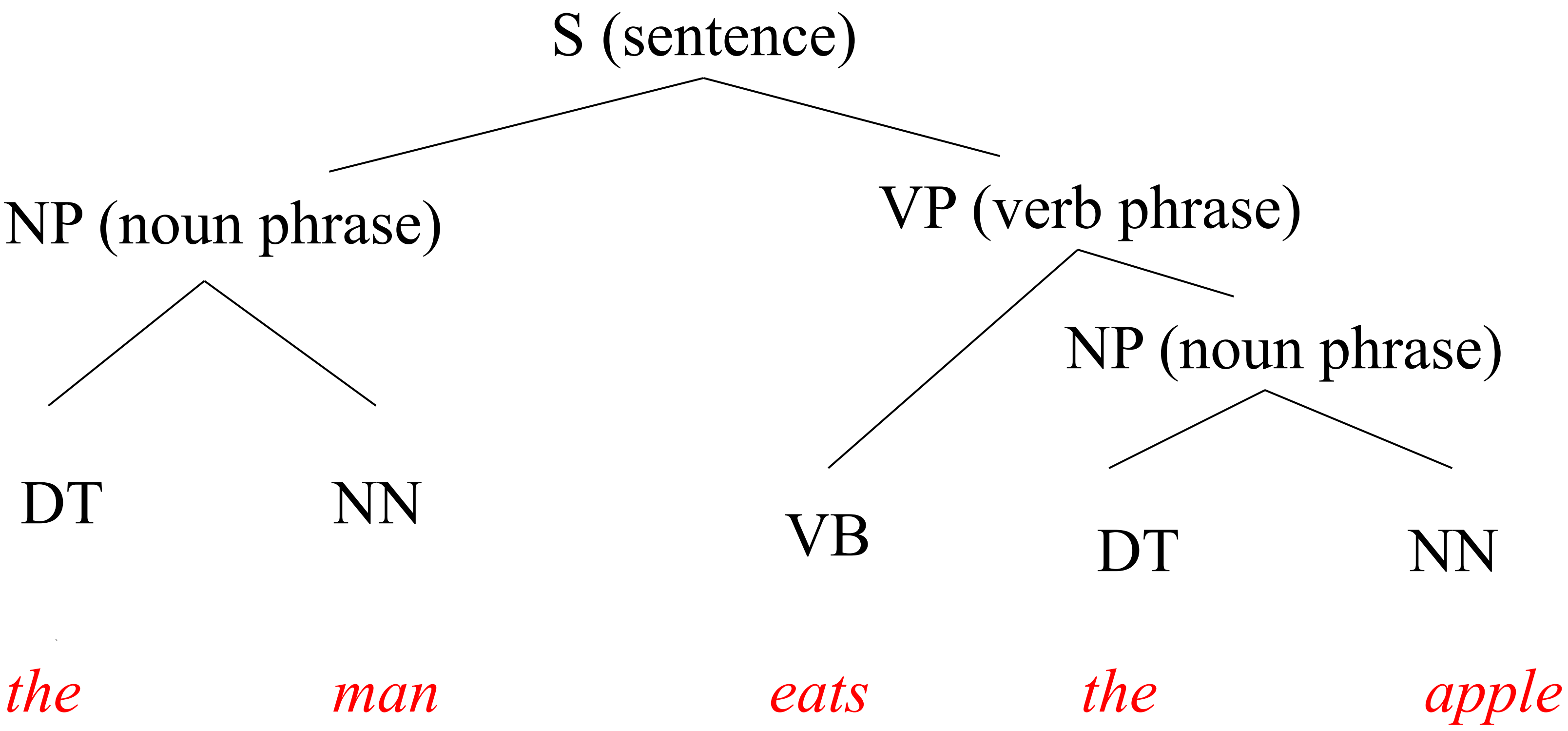


Context Free Grammar Rules:

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



S (sentence)

NP (noun phrase)

VP (verb phrase)

NP (noun phrase)

DT NN VB DT NN

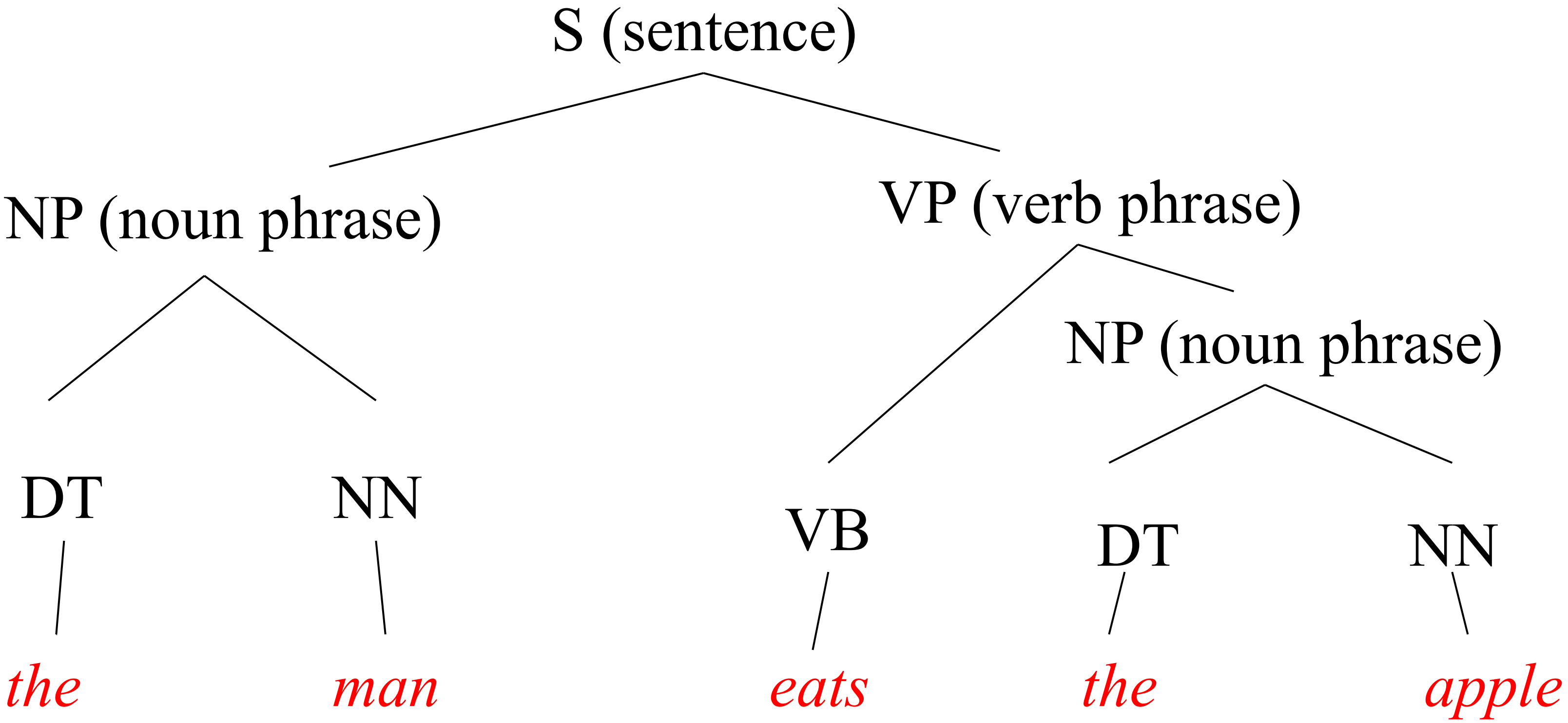
the *man* *eats* *the* *apple*

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 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.

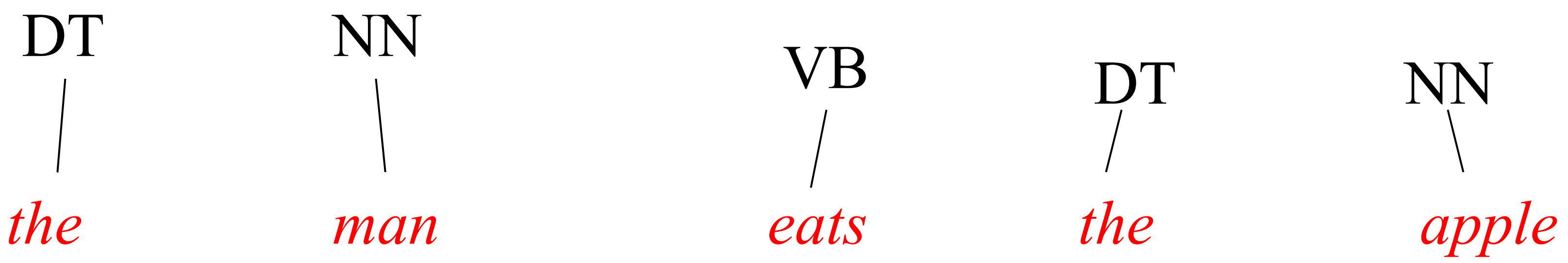
the *man* *eats* *the* *apple*

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

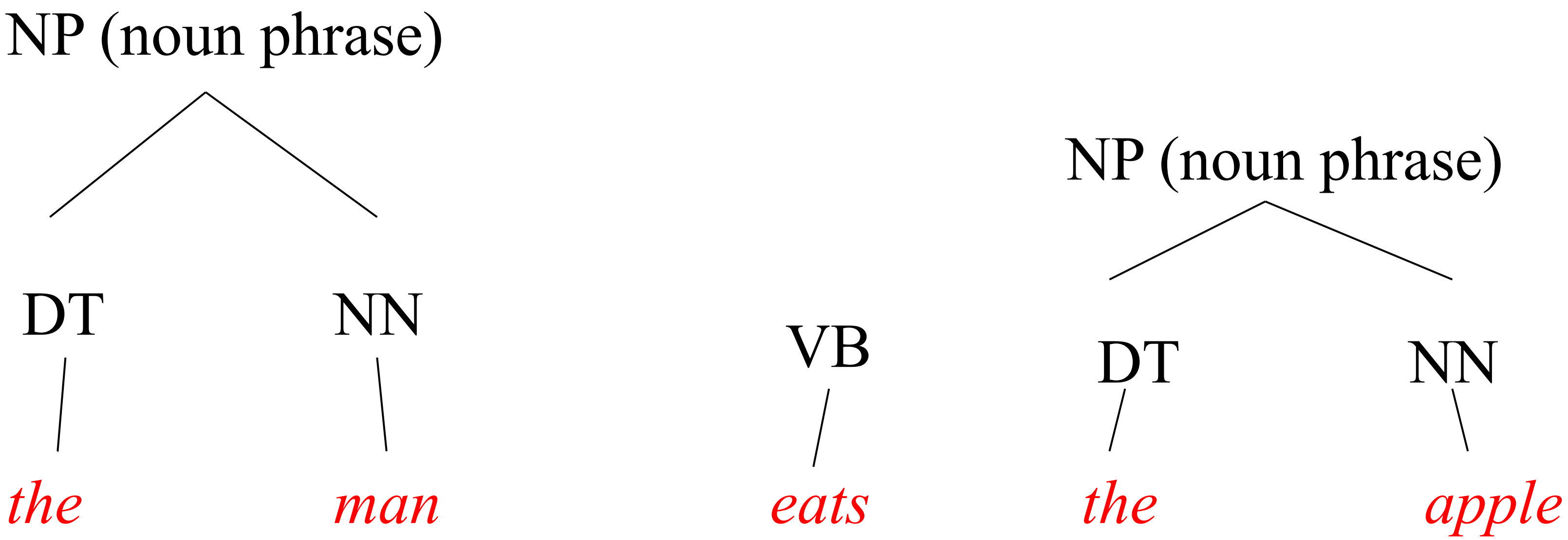


Context Free Grammar Rules:

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



NP (noun phrase)

DT

NN

the

man

VB

eats

NP (noun phrase)

DT

NN

the

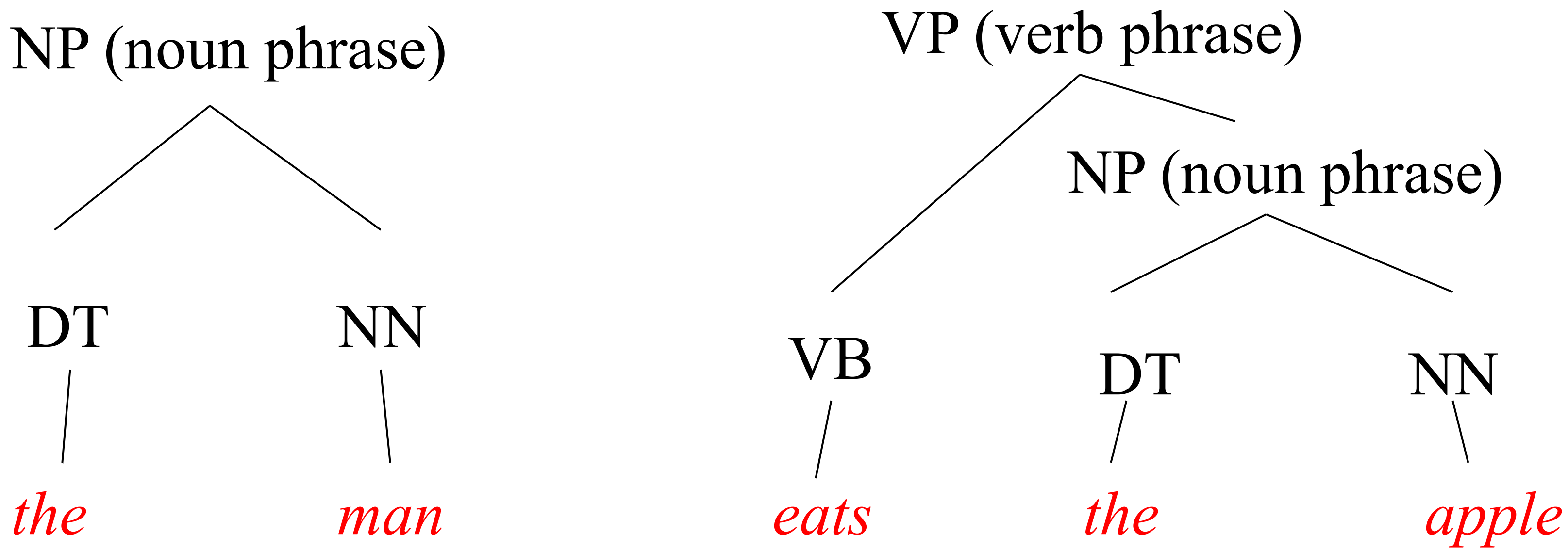
apple

Context Free Grammar Rules:

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

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

Bottom Up Derivation – add VP rule



NP (noun phrase)

DT NN

the *man*

VP (verb phrase)

VB NP (noun phrase)

DT NN

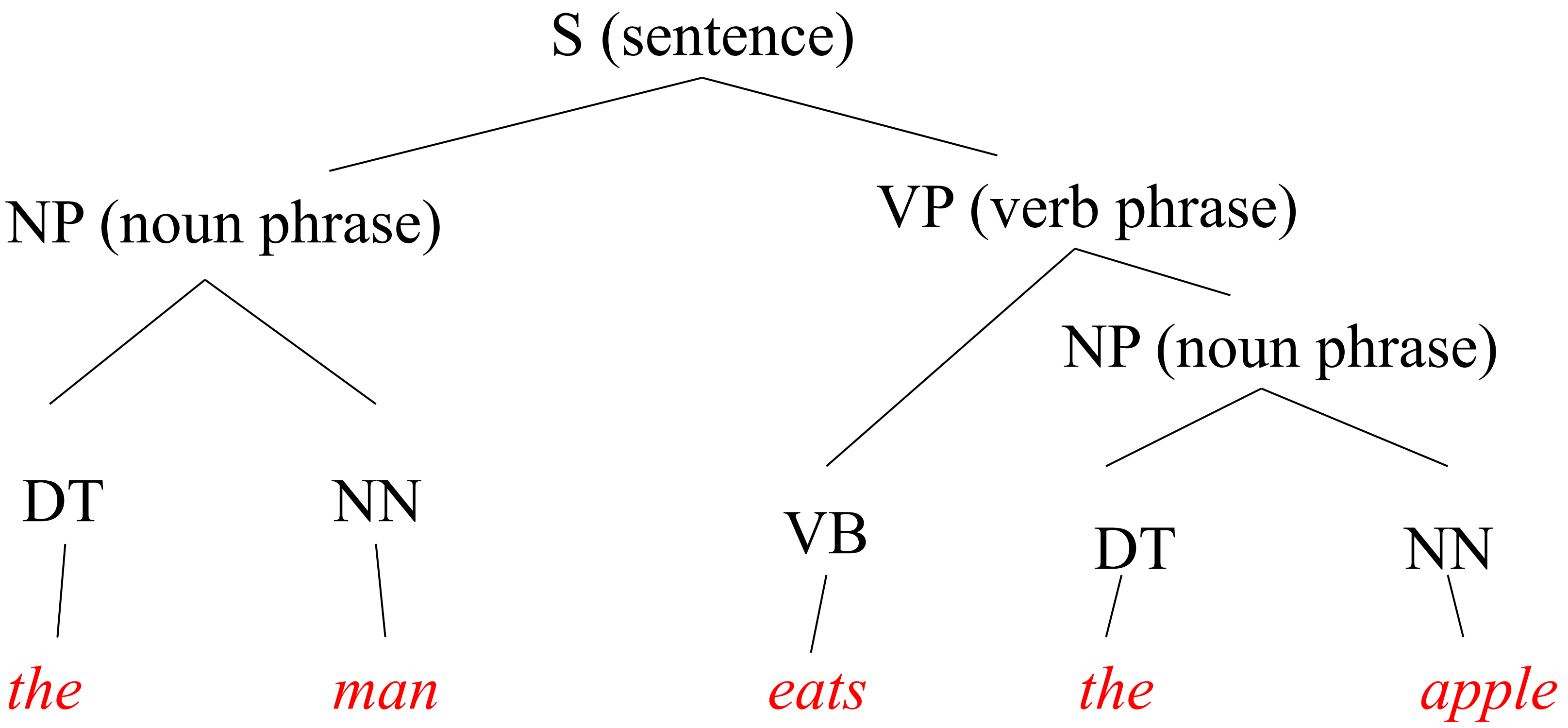
eats *the* *apple*

Context Free Grammar Rules:

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

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

Bottom Up Derivation – add S rule



S (sentence)

NP (noun phrase)

VP (verb phrase)

NP (noun phrase)

DT NN VB DT NN

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* | …

Notations for (constituent) syntactic structure

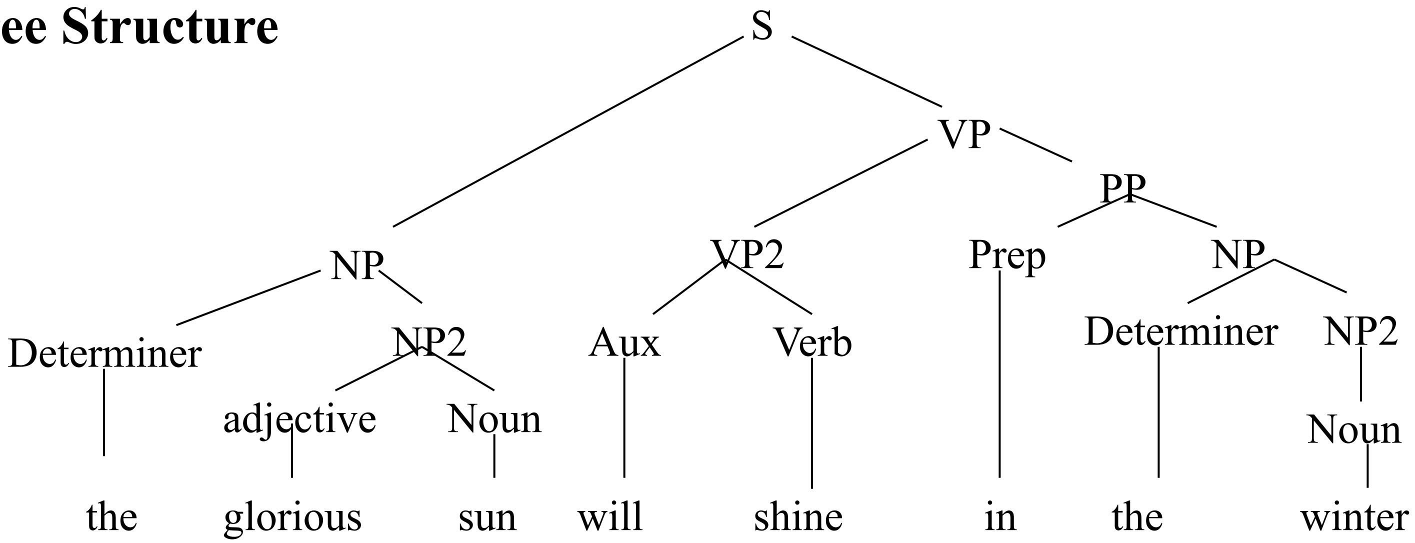
Bracketed text

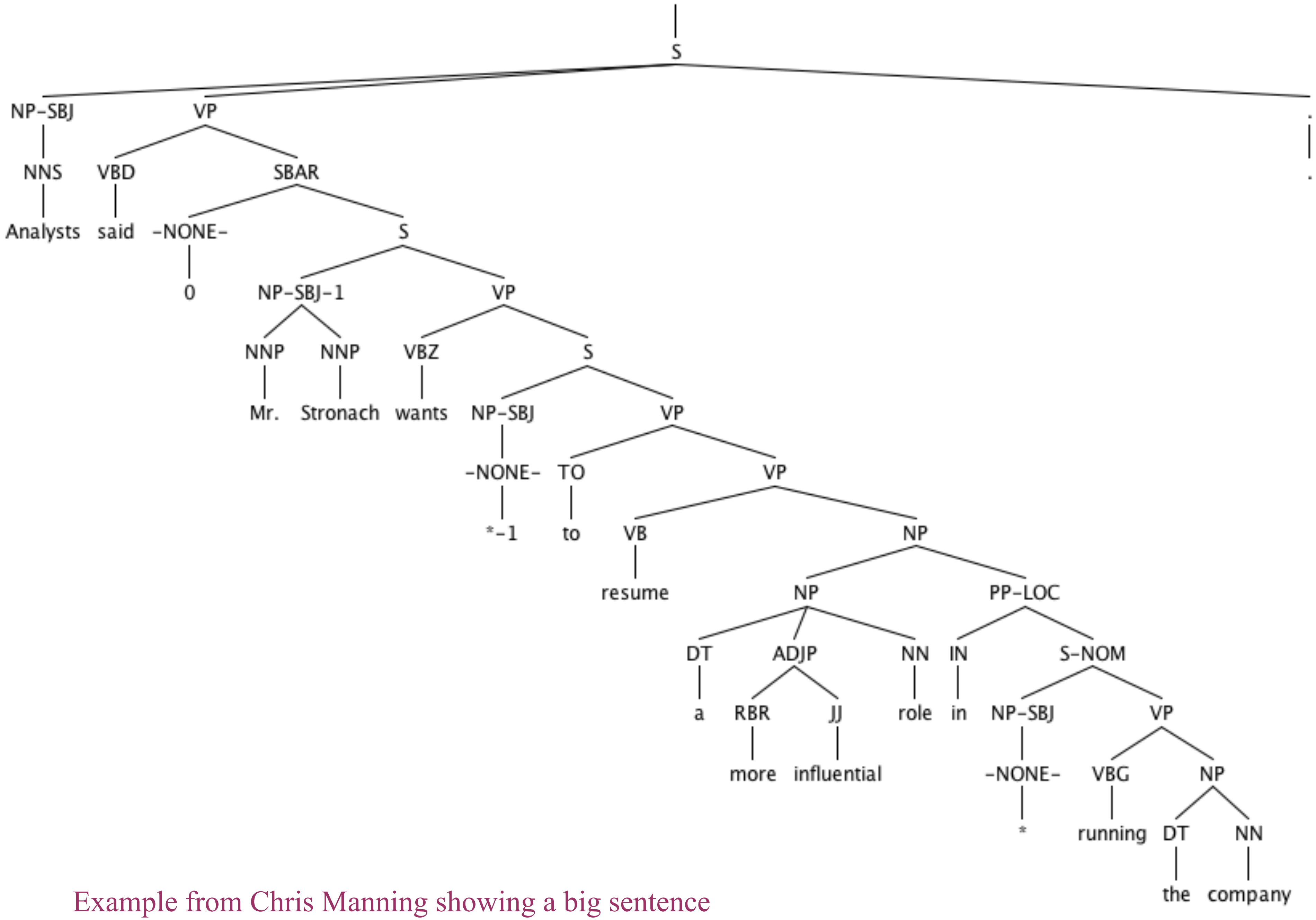
[$_S$ [$_{NP}$ the [$_{NP2}$ glorious sun]] [$_{VP}$ [$_{VP2}$ will shine] [$_{PP}$ in [$_{NP}$ the [$_{NP2}$ winter]]]]]

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

Context-Free Grammars

- Why Context-Free?
  - The notion of context in CFGs has nothing to do with the ordinary meaning of the word context in language.
  - All it really means is that the non-terminal on the left-hand side of a rule can be replaced regardless of context
    - Context-sensitive grammars allow context to be placed on the left-hand side of the rewrite rule
- In programming languages, and other uses of CFGs in Computer Science, notably XML, CFGS are
  - Unambiguous
    - Assign at most, 1 structural description to a string
  - Parsable in time linearly proportional to the length of the string