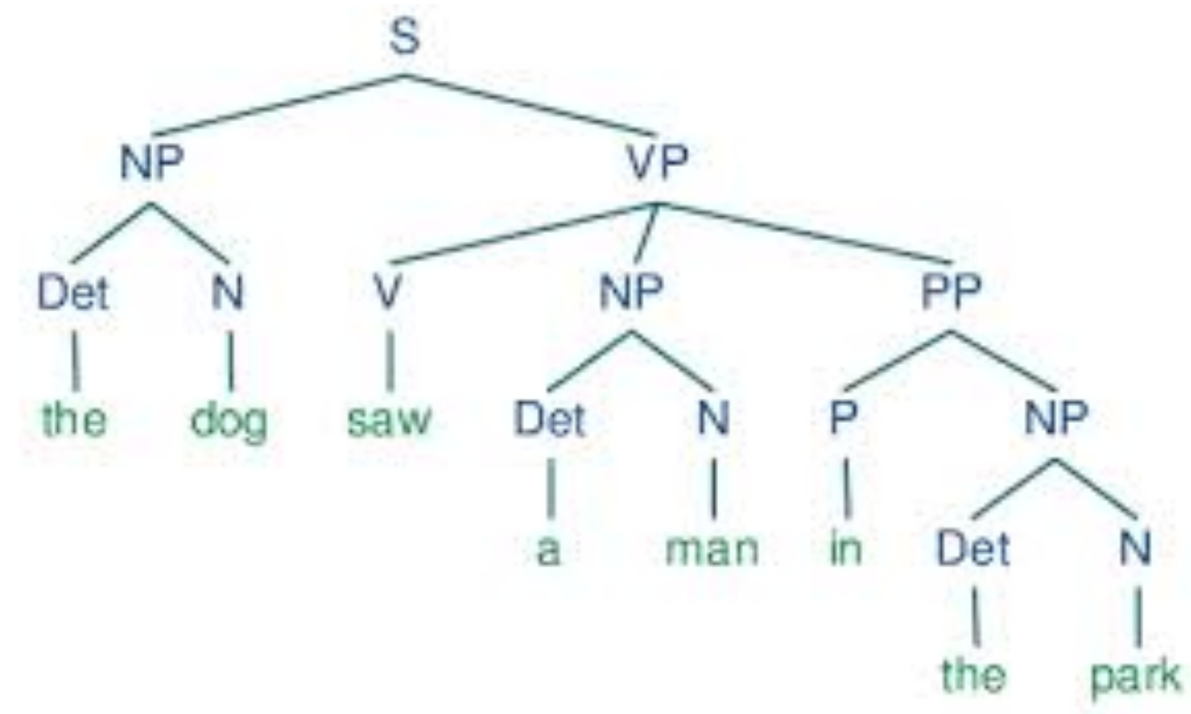


COMP90042 LECTURE 8

CONTEXT-FREE GRAMMARS

SYNTACTIC CONSTITUENTS

- Sequential models like HMMs assume entirely flat structure
- But language clearly isn't like that

[*A man*] [*saw* [*a dog*] [*in* [*the park*]]]

- Words group together to form syntactic constituents
  - Can be replaced, or moved around *as a unit*
- Grammars allow us to formalize these intuitions
  - Symbols correspond to syntactic constituents

TESTING FOR CONSTITUENCY

- Various tests for constituency, based on linguistic intuition
  - *Only constituents can answer a question*

Trevor gave a lecture on grammar	*Who gave the lecture on grammar?*	*Trevor*
Trevor gave a lecture on grammar	*Trevor did what with the lecture on grammar?*	**gave (fails)*
Trevor gave a lecture on grammar	*What topic was Trevor's lecture on?*	*on grammar*

  - *Only constituents can be coordinated with others* (*of same type*)

    Trevor gave a lecture on grammar and on parsing
    Trevor gave a lecture on grammar and parsing
    Trevor gave a lecture on grammar and a treatise on parsing

    Trevor gave a lecture on grammar and ate a tasty pie

    #Trevor gave a lecture on and a treatise about grammar

- More tests, e.g., *topicalisation*, *clefting* and *coordination*.

OUTLINE

- The context-free grammar formalism
- Parsing with CFGs
- Representing English with CFGs

BASICS OF CONTEXT-FREE GRAMMARS

- Symbols
  - Terminal: word such as *book*
  - Non-terminal: syntactic label such as NP or NN
  - Convention to use upper and lower-case to distinguish, or else "quotes" for terminals
- Productions (rules)

$$W \rightarrow X\ Y\ Z$$

  - Exactly one non-terminal on left-hand side (LHS)
  - An ordered list of symbols on right-hand side (RHS) — can be Terminals or Non-terminals

REGULAR EXPRESSIONS AS CFGS

- Regular expressions match simple patterns
  - E.g., [A-Z][a-z]*   words starting with a capital
- Can rewrite as a grammar
  - $S \rightarrow C$ $S \rightarrow C\ RS$
  - $C \rightarrow$ "A" $C \rightarrow$ "B"   … $C \rightarrow$ "Z"
  - $RS \rightarrow R$ $RS \rightarrow R\ RS$
  - $R \rightarrow$ "a" $R \rightarrow$ "b"   … $R \rightarrow$ "z"
- In fact, a regex is a way of specifying **a regular language** (*language* = set of strings matching pattern)
  - The class of regular languages is a subset of the **context-free languages**, which are specified using a CFG

CFGS VS REGULAR GRAMMARS

- CFGs (and regexs) used to describe a set of strings, aka a "*language*"
- Regular grammars
  - describe a smaller class of languages, e.g., a*b*c*
  - can be parsed using finite state machine
  - HMMs are a model for a (weighted) regular grammar (*exercise: show the grammar for a POS tagging HMM*)
- CFGs
  - can describe hierarchical groupings e.g., matching brackets ($a^n b^n$) & many recursive tree structures found in language
  - requires push-down automata to parse
- Context sensitive grammars are even more expressive (and intractable)

A SIMPLE GRAMMAR

Terminal symbols: *rat, the, ate, cheese*

Non-terminal symbols: S, NP, VP, DT, VBD, NN

Productions:

S $\rightarrow$ NP VP

NP $\rightarrow$ DT NN

VP $\rightarrow$ VBD NP

DT $\rightarrow$ *the*

NN $\rightarrow$ *rat*

NN $\rightarrow$ *cheese*

VBD $\rightarrow$ *ate*

GENERATING SENTENCES WITH CFGS

Always start with S (the sentence/start symbol)

S

Apply rule with S on LHS (S $\rightarrow$ NP VP), i.e substitute RHS

NP VP

Apply rule with NP on LHS (NP $\rightarrow$ DT NN)

DT NN VP

Apply rule with DT on LHS (DT $\rightarrow$ *the*)

***the* NN VP**

Apply rule with NN on LHS (NN $\rightarrow$ *rat*)

***the rat* VP**

GENERATING SENTENCES WITH CFGS

Apply rule with VP on LHS (VP $\rightarrow$ VBD NP)

***the rat* VBD NP**

Apply rule with VBD on LHS (VBD $\rightarrow$ *ate*)

***the rat ate* NP**

Apply rule with NP on LHS (NP $\rightarrow$ DT NN)

***the rat ate* DT NN**

Apply rule with DT on LHS (DT $\rightarrow$ *the*)

***the rat ate the* NN**

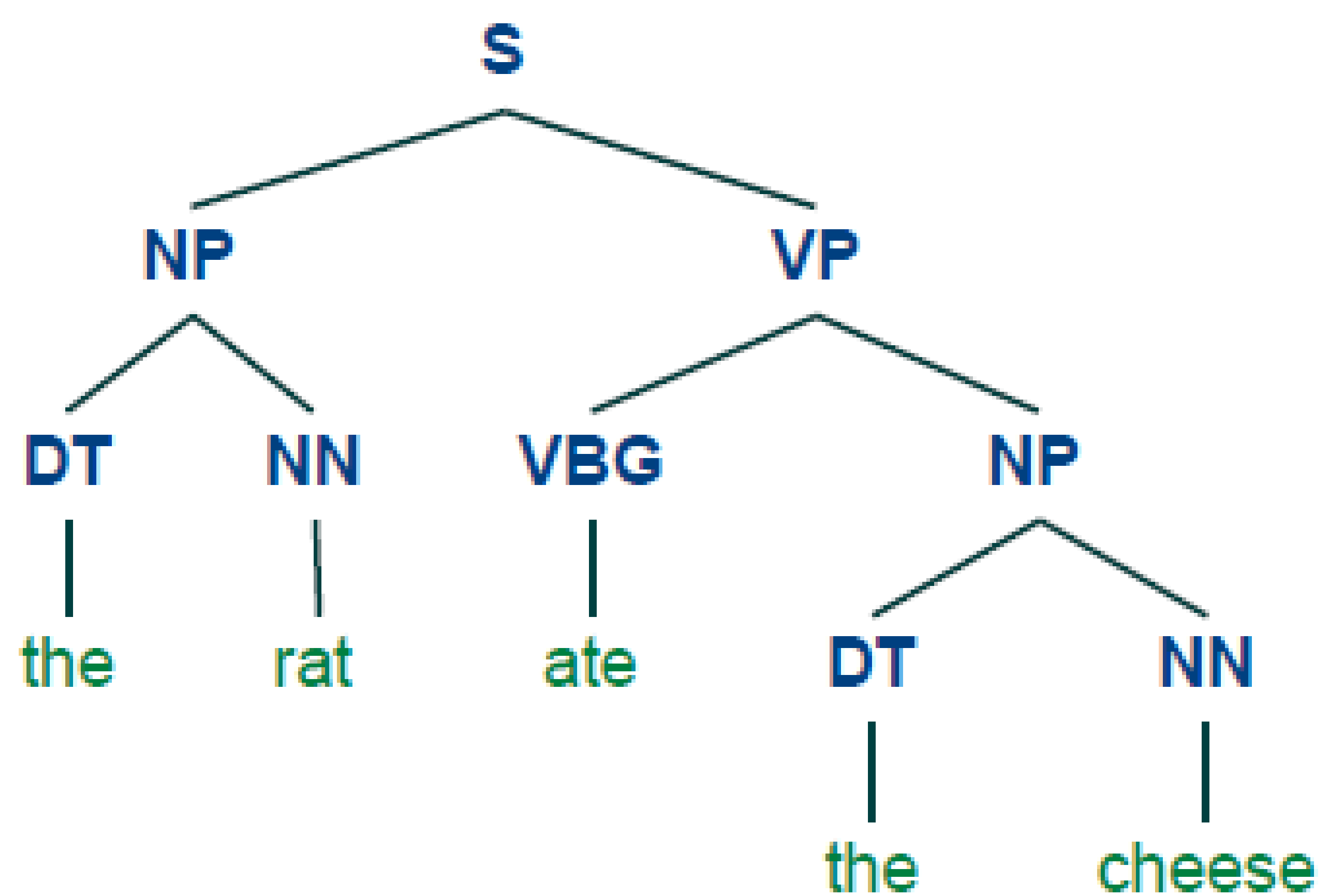
Apply rule with NN on LHS (NN $\rightarrow$ *cheese*)

the rat ate the cheese

CFG TREES

- Generation corresponds to a syntactic tree
- Non-terminals are internal nodes
- Terminals are leaves

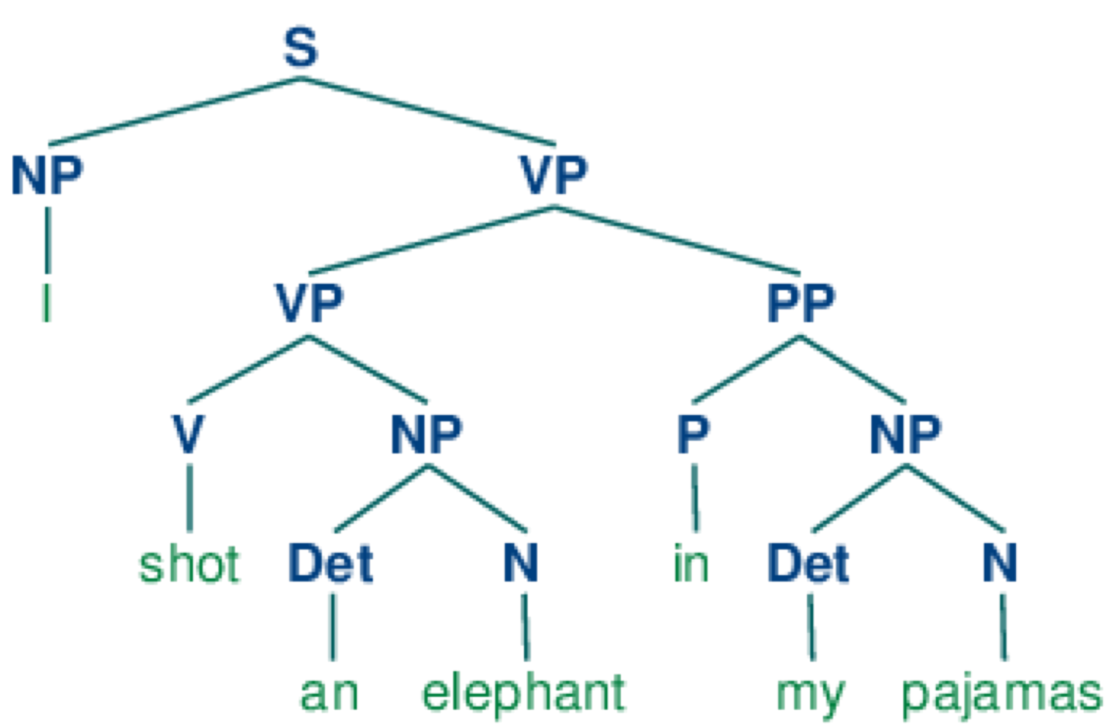
```
(S (NP (DT the)
       (NN rat))
   (VP (VBG ate)
       (NP (DT the)
           (NN cheese))))
```

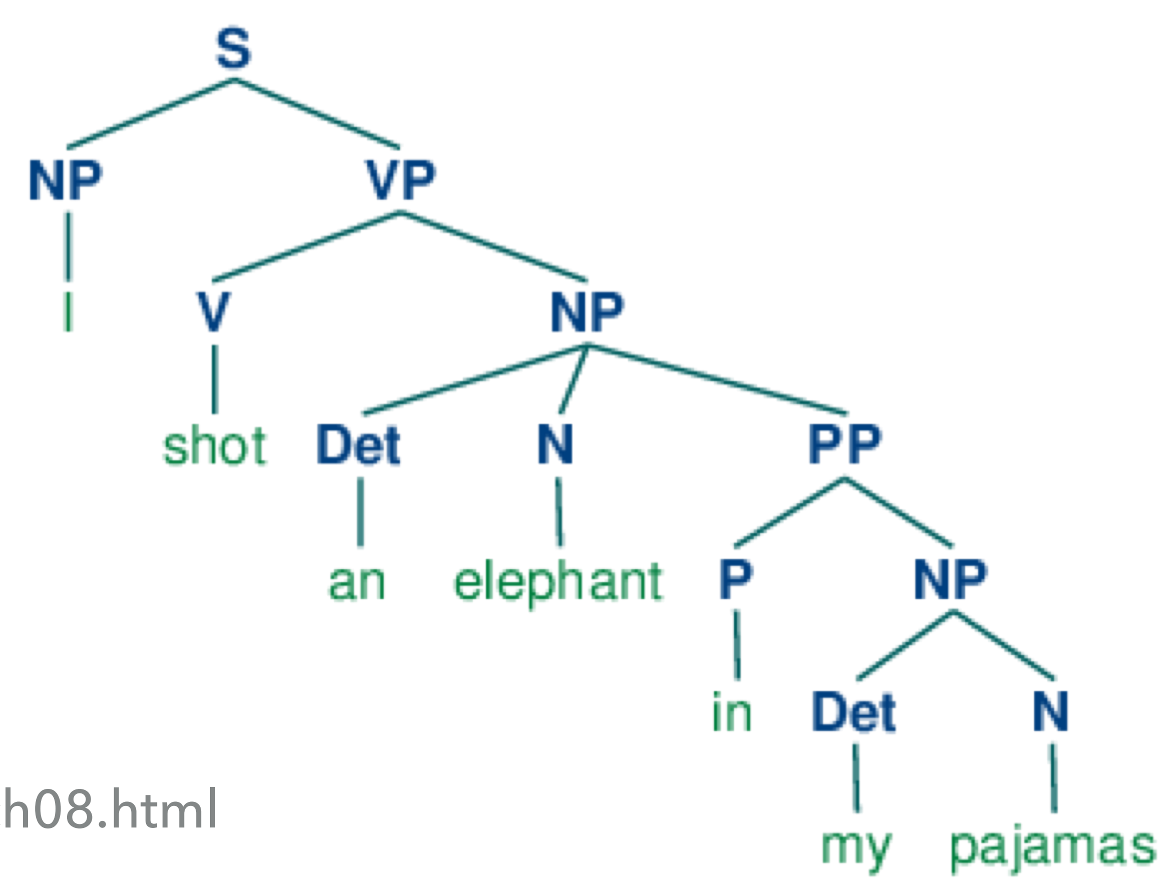


- Parsing is the reverse process

PARSE AMBIGUITY

- Often more than one tree can describe a string
- "While hunting in Africa, I shot an elephant in my pajamas. How he got into my pajamas, I don't know." *Animal Crackers* (1930)





Example & figures: http://www.nltk.org/book/ch08.html

PARSING CFGS

- Parsing: given string, identify possible structures
- Brute force search is intractable for non-trivial grammars
  - Good solutions use dynamic programming
- Two general strategies
  - Bottom-up
    - Start with words, work up towards S
    - CYK parsing
  - Top-down
    - Start with S, work down towards words
    - Earley parsing (not covered)

THE CYK PARSING ALGORITHM

- Convert grammar to Chomsky Normal Form (CNF)
- Fill in a parse table
- Use table to derive parse
- Convert result back to original grammar

CONVERT TO CNF

- Change grammar so all rules of form
  $A \rightarrow B\ C$ or $A \rightarrow a$
- Step 1: Convert rules of form
  $A \rightarrow B\ c$ into pair of rules $A \rightarrow B\ X$, $X \rightarrow c$
  - Not usually necessary in POS-based grammars
- Step 2: Convert rules $A \rightarrow B\ C\ D$ into $A \rightarrow B\ Y$, $Y \rightarrow C\ D$
  - Usually necessary, but not for our toy grammar
- *X*, *Y* are new symbols we have introduced
- Unary rules $A \rightarrow B$ can be permitted, but have to factor out cycles)

PARSE TABLE

the	*rat*	*ate*	*the*	*cheese*
DT [0,1]	NP [0,2]	[0,3]	[0,4]	S [0,5]
	NN [1,2]	[1,3]	[1,4]	[1,5]
		VBD [2,3]	[2,4]	VP [2,5]
			DT [3,4]	NP [3,5]
				NN [4,5]

S → NP VP

NP → DT NN

VP → VBD NP

DT → *the*

NN → *rat*

NN → *cheese*

VBD → *ate*

CYK ALGORITHM

function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

Figure 12.5 The CKY algorithm.

- JM3, Ch 12

CYK: RETRIEVING THE PARSES

- S in the top-left corner of parse table indicates success
- To get parse(s), follow pointers back for each match
- Convert back from CNF by removing new non-terminals

PARSE TABLE WITH BACKPOINTERS

the	*rat*	*ate*	*the*	*cheese*
DT [0,1]	NP *Split = 1;* *NP → DT NN* [0,2]	[0,3]	[0,4]	S *Split = 2;* *S → NP VP* [0,5]
	NN [1,2]	[1,3]	[1,4]	[1,5]
		VBD [2,3]	[2,4]	VP *Split = 3;* *VP → VBD NP* [2,5]
			DT [3,4]	NP *Split = 4;* *NP → DT NN* [3,5]
				NN [4,5]

S → NP VP

NP → DT NN

VP → VBD NP

DT → *the*

NN → *rat*

NN → *cheese*

VBD → *ate*

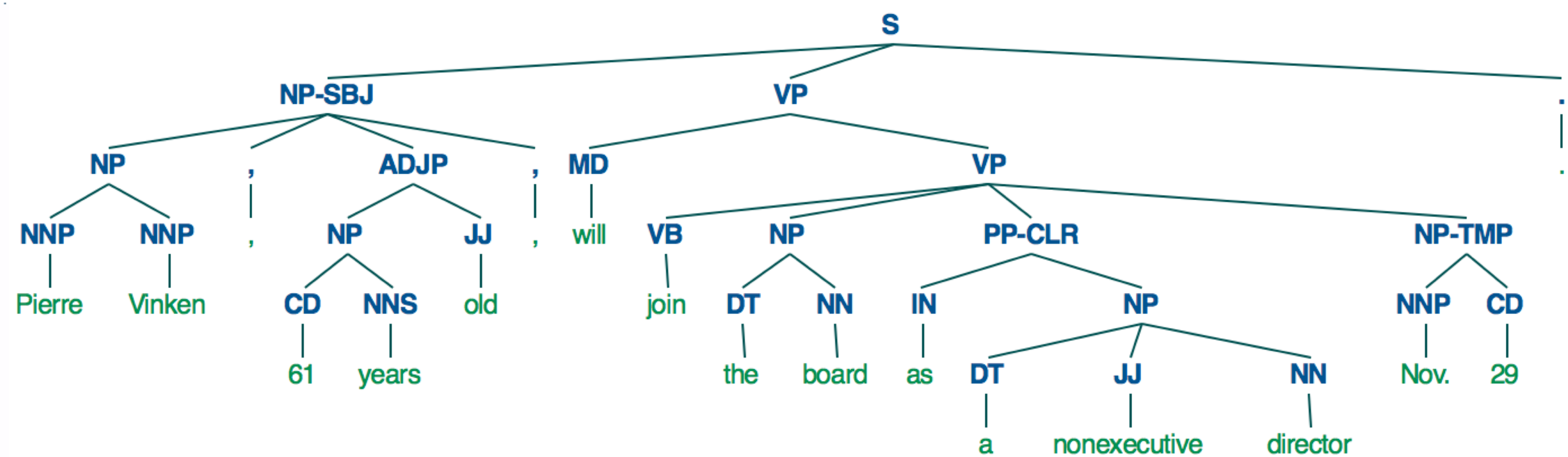
FROM TOY GRAMMARS TO REAL GRAMMARS

- Toy grammars with handful of productions good for demonstration or extremely limited domains
- For real texts, we need real grammars
- Hundreds or thousands of production rules

KEY CONSTITUENTS IN PENN TREEBANK

- Sentence (S)
- Noun phrase (NP)
- Verb phrase (VP)
- Prepositional phrase (PP)
- Adjective phrase (AdjP)
- Adverbial phrase (AdvP)
- Subordinate clause (SBAR)

EXAMPLE PTB/0001




```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

BASIC ENGLISH SENTENCE STRUCTURES

- Declarative sentences (S $\rightarrow$ NP VP)
  - E.g. *The rat ate the cheese*
- Imperative sentences (S $\rightarrow$ VP)
  - E.g. *Eat the cheese!*
- Yes/no questions (S $\rightarrow$ VB NP VP)
  - E.g. *did the rat eat the cheese?*
- *Wh*-subject-questions (S $\rightarrow$ WH VP)
  - *Who ate the cheese?*
- *Wh*-object-questions (S $\rightarrow$ WH VB NP VP)
  - *What did the rat eat?*

ENGLISH NOUN PHRASES

- Pre-modifiers
  - DT, CD, ADJP, NNP, NN
  - E.g. *the two very best Philly cheese steaks*
- Post-modifiers
  - PP, VP, SBAR
  - A delivery from Bob coming today that I don't want to miss

NP → DT? CD? ADJP? (NN | NNP)+ PP* VP? SBAR?

NP → PRP

VERB PHRASES

- Auxiliaries
  - MD, AdvP, VB, TO
  - E.g *should really have tried to wait*

VP → (MD | VB | TO) AdvP? VP

- Arguments and adjuncts
  - NP, PP, SBAR, VP, AdvP
  - E.g *told him yesterday that I was ready*
  - E.g. *gave John a gift for his birthday to make amends*

VP → VB NP? NP? PP* AdvP* VP? SBAR?

OTHER CONSTITUENTS

- Prepositional phrase
  - PP $\rightarrow$ IN NP (*in the house*)
- Adjective phrase
  - AdjP $\rightarrow$ (AdvP) JJ (*really nice*)
- Adverb phrase
  - AdvP $\rightarrow$ (AdvP) RB (*not too well*)
- Subordinate clause
  - SBAR $\rightarrow$ (IN) S (*since I came here*)
- Coordination
  - NP $\rightarrow$ NP CC NP; VP $\rightarrow$ VP CC VP; etc. (*Jack and Jill*)
- Complex sentences
  - S $\rightarrow$ S SBAR; S $\rightarrow$ SBAR , S; etc. (*if he goes, I'll go*)

A FINAL WORD

- Context-free grammars can represent linguistic structure
- There are relatively fast dynamic programming algorithms to retrieve this structure
- But what about ambiguity?
  - Extreme ambiguity will slow down parsing
  - If multiple possible parses, which is best?

REQUIRED READING

- J&M3 Ch. 11.1-11.5, Ch. 12.1-12.2
- Constituency tests http://people.umass.edu/nconstan/201/Constituency%20Tests.pdf