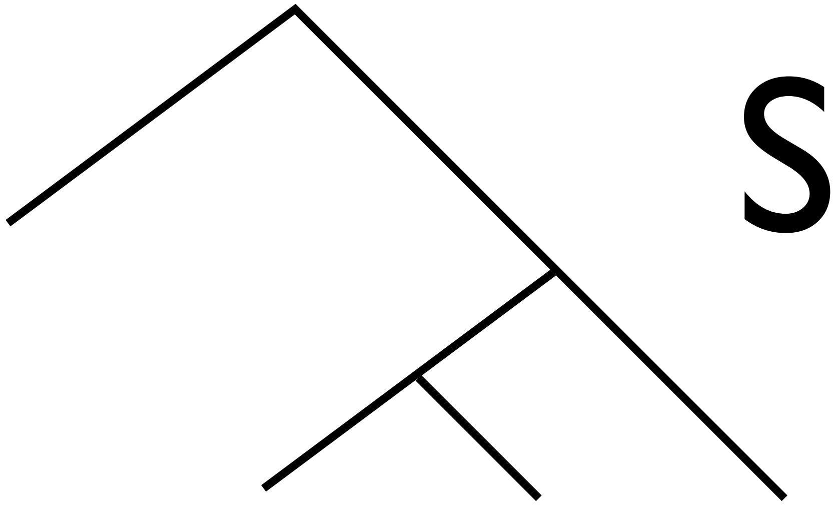
CS 544 NLP
Spring 2011

S → NP VP

Syntax and Parsing

Dirk Hovy
1-13-2011
(some slides from Liang Huang)

CS 544 NLP
Spring 2011



S → NP VP

Syntax and Parsing

Dirk Hovy
1-13-2011
(some slides from Liang Huang)

CS 544 NLP
Spring 2011

S → NP VP

Syntax and Parsing

Dirk Hovy
1-13-2011
(some slides from Liang Huang)

What’s wrong here?

hovercraft full my is eels of

What’s wrong here?

my hovercraft is full of eels

Order, please!

- some orders are grammatical, others not

**hovercraft full my is eels of*

vs.

my hovercraft is full of eels

Order, please!

- some orders are grammatical, others not

**hovercraft full my is eels of*

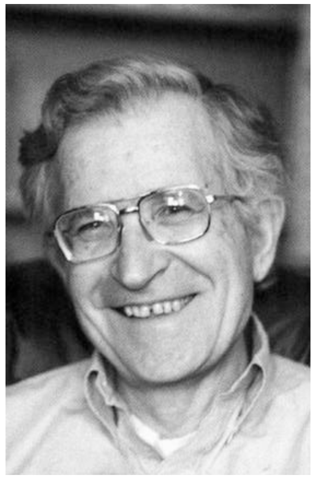
vs.

my hovercraft is full of eels

ungrammatical sentences are marked with a *

Syntax

- study of word order
- one of fundamental levels of language (phonetics/phonology, morphology, syntax, semantics, pragmatics)
- has to do with trees…
- Chomsky says: independent of meaning!
  Colorless green ideas sleep furiously



Sentence elements

- *[my hovercraft] [is full of eels]*
  [it] [is full of eels]
  [my air-powered aquatic vehicle] [is full of eels]
  [my hovercraft] [sank]
- we can exchange certain elements: phrases (or constituents)

How to spot phrases from a large distance

- substitution:
  it is full of eels {*it* = *my hovercraft*}
- deletion (produces nonsense):
  **Ø is full of eels*

Recurring structures

- substitution shows: many sentences have the same structure
- pick any two to make a sentence:

my hovercraft
Dennis Moore
a man with three buttocks

is full of eels
has a brother
owns a shack
is huge

Recurring structures

- substitution shows: many sentences have the same structure
- pick any two to make a sentence:

a ma

Noun phrases

ks

is full of eels
has a brother
owns a shack
is huge

Recurring structures

- substitution shows: many sentences have the same structure
- pick any two to make a sentence:

Noun phrases

Verb phrases

a ma

ks

slide by Liang Huang

Context-Free Grammars

- S → **NP VP**
- **NP → Det N**
- **NP → NP PP**
- PP → P NP
- VP → V NP
- VP → VP PP
- ...

- **N →** *{ball, garden, house, sushi }*
- **P →** *{in, behind, with}*
- **V → ...**
- **Det → ...**

slide by Liang Huang

Context-Free Grammars

- S → **NP VP**
- **NP → Det N**
- **NP PP**
- PP → P NP
- VP → V NP
- VP → VP PP
- ...

most famous rule in linguistics ever…

- **N →** *{ball, garden, house, sushi }*
- *behind, with}*
- **V → ...**
- **Det → ...**

slide by Liang Huang

Context-Free Grammars

A CFG is a 4-tuple 〈N,Σ,R,S〉

A set of nonterminals N
(e.g. N = {S, NP, VP, PP, Noun, Verb, ...})

A set of terminals Σ
(e.g. Σ = {*I, you, he, eat, drink, sushi, ball,* })

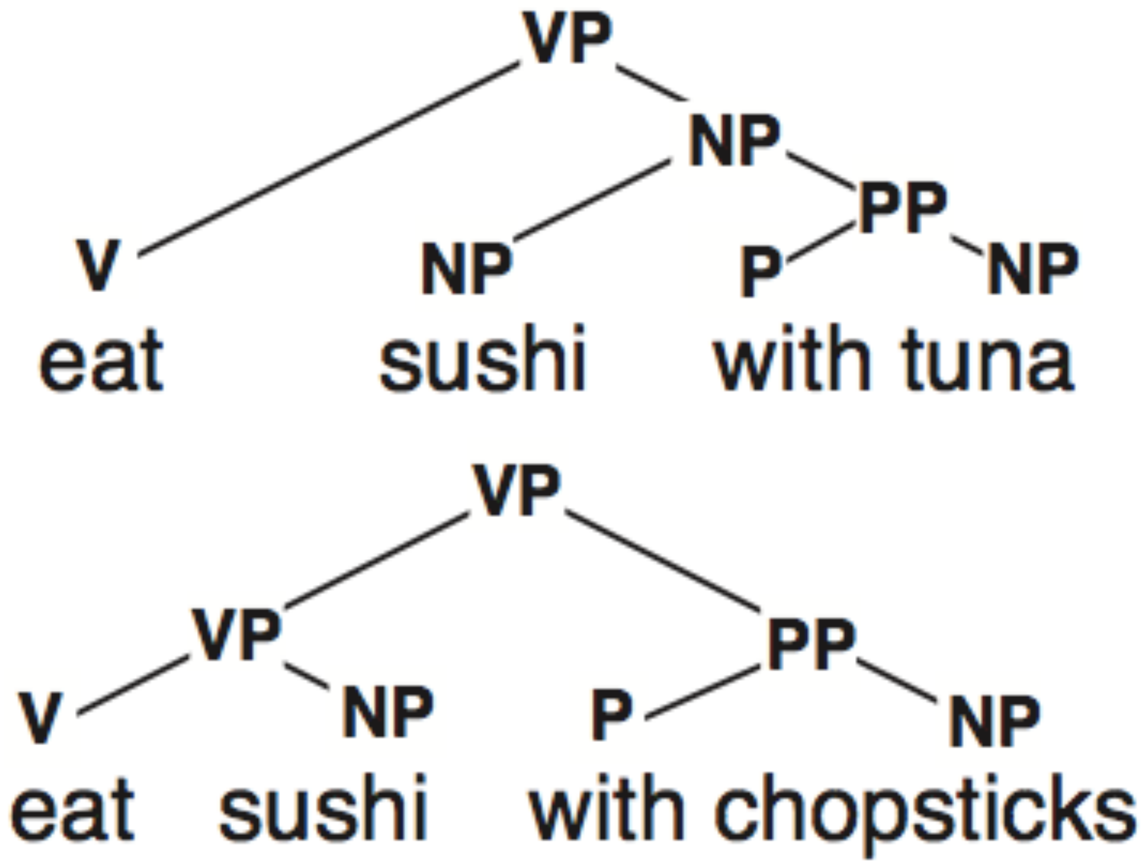
A set of rules R
R ⊆ **{**A → β **with left-hand-side (LHS)** A ∈ N
and right-hand-side (RHS) β ∈ **(N** ∪ Σ)* **}**

A start symbol S (sentence)

slide by Liang Huang

Parse Trees

- **N → *{sushi, tuna}***
- **P → *{with}***
- **V → *{eat}***
- **NP → N**
- NP → NP PP
- PP→P NP
- VP→V NP
- VP→VP PP

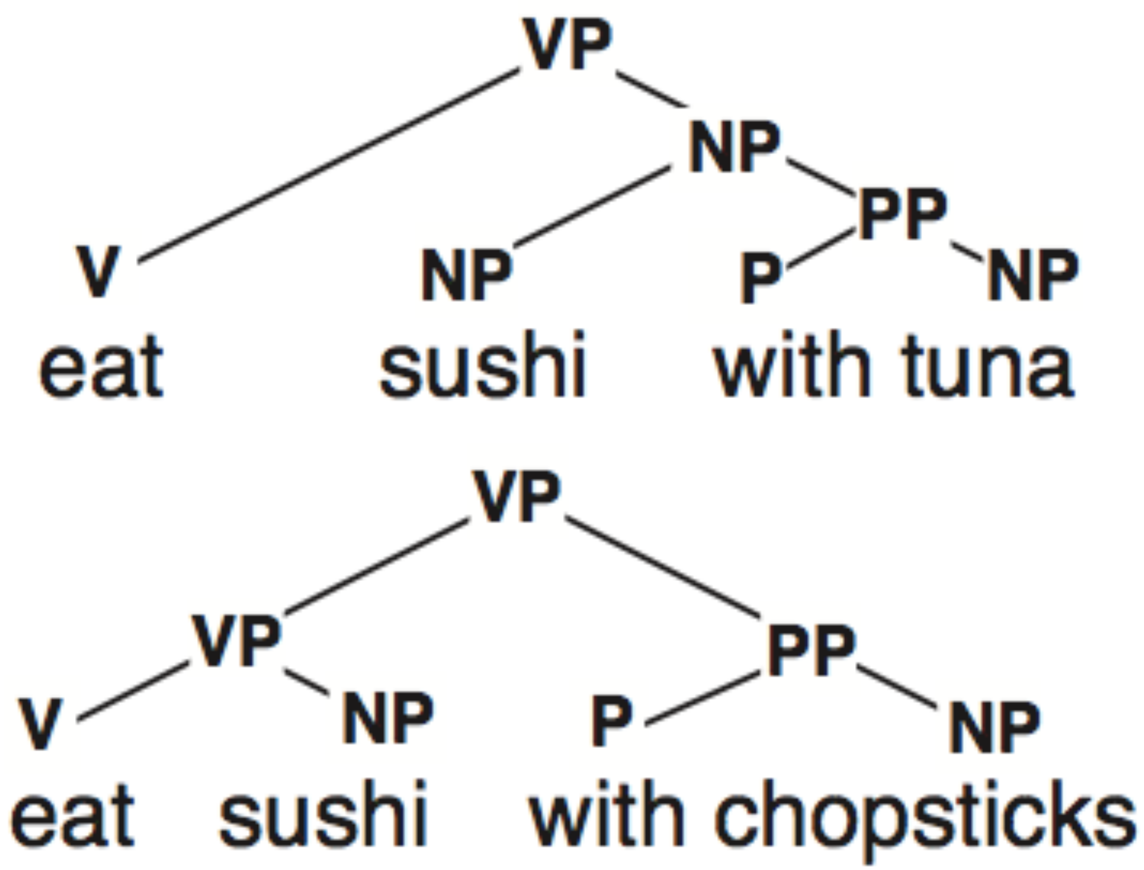


slide by Liang Huang

Parse Trees

terminals

- **N → *{sushi, tuna}***
- **P → *{with}***
- **V → *{eat}***
- **NP → N**
- NP → NP PP
- PP→P NP
- VP→V NP
- VP→VP PP

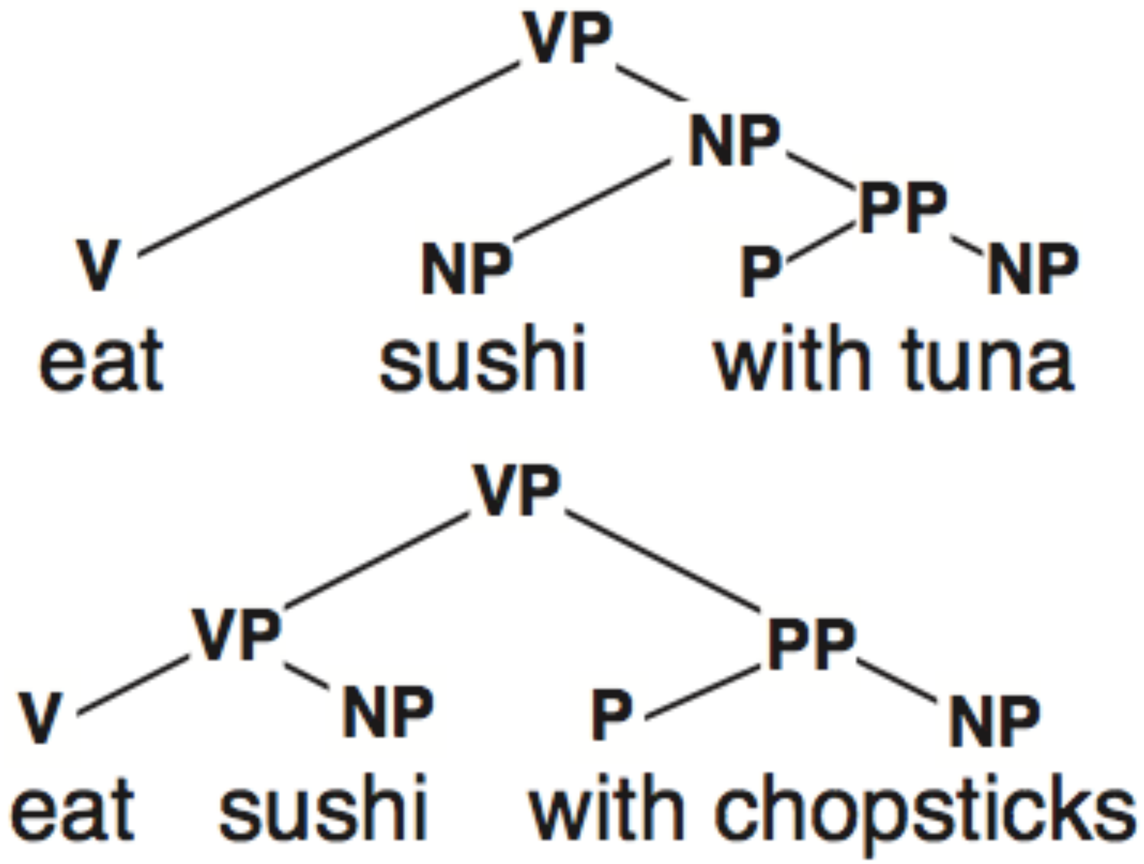


Parse Trees

pre-terminals

terminals

- **N** → *{sushi, tuna}*
- **P** → *{with}*
- **V** → *{eat}*
- **NP → N**
- NP → NP PP
- PP→P NP
- VP→V NP
- VP→VP PP



slide by Liang Huang

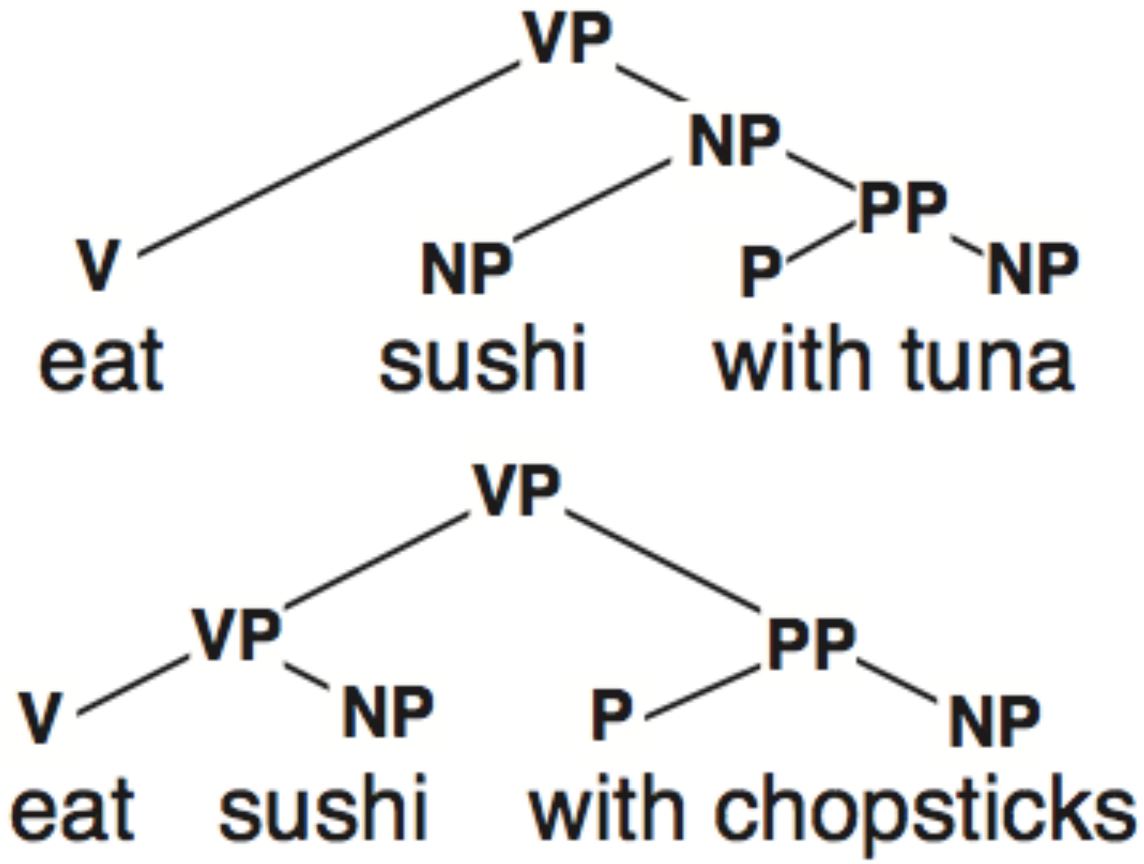
Parse Trees

pre-terminals

terminals

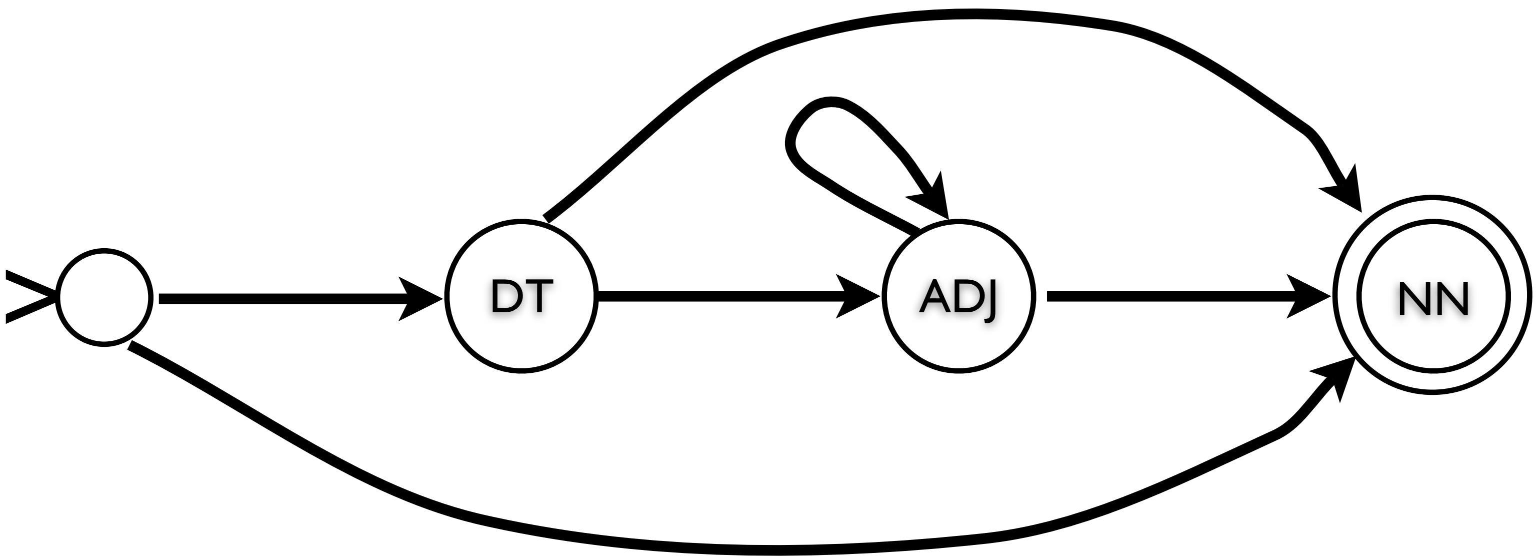
- **N** → *{sushi, tuna}*
- **P** → *{with}*
- **V** → *{eat}*
- **NP** → **N**
- NP → NP PP
- PP → P NP
- VP → V NP
- VP → VP PP

non-terminals



Grammaticality

- a sentence is grammatical if there is an acceptor for it



Generate from CFGs

```
initialize stack with S
while stack not empty:
    x = stack.pop()
    if x ∈ terminals:
      print x
    else if x ∈ rule:
      stack.push(y in RHS for selected x → RHS)
```

Parsing

- find a path b/w root node S and terminals
- recursively apply CFG rules
- glorified search
- options:
  - direction: top-down, bottom-up
  - expansion: breadth-first, depth-first, bidirectional

Probabilistic parsing

- some rules are more likely than others:
  N → dog, 0.9
  V → dog, 0.1
- use probabilities to decide best path

Playtime

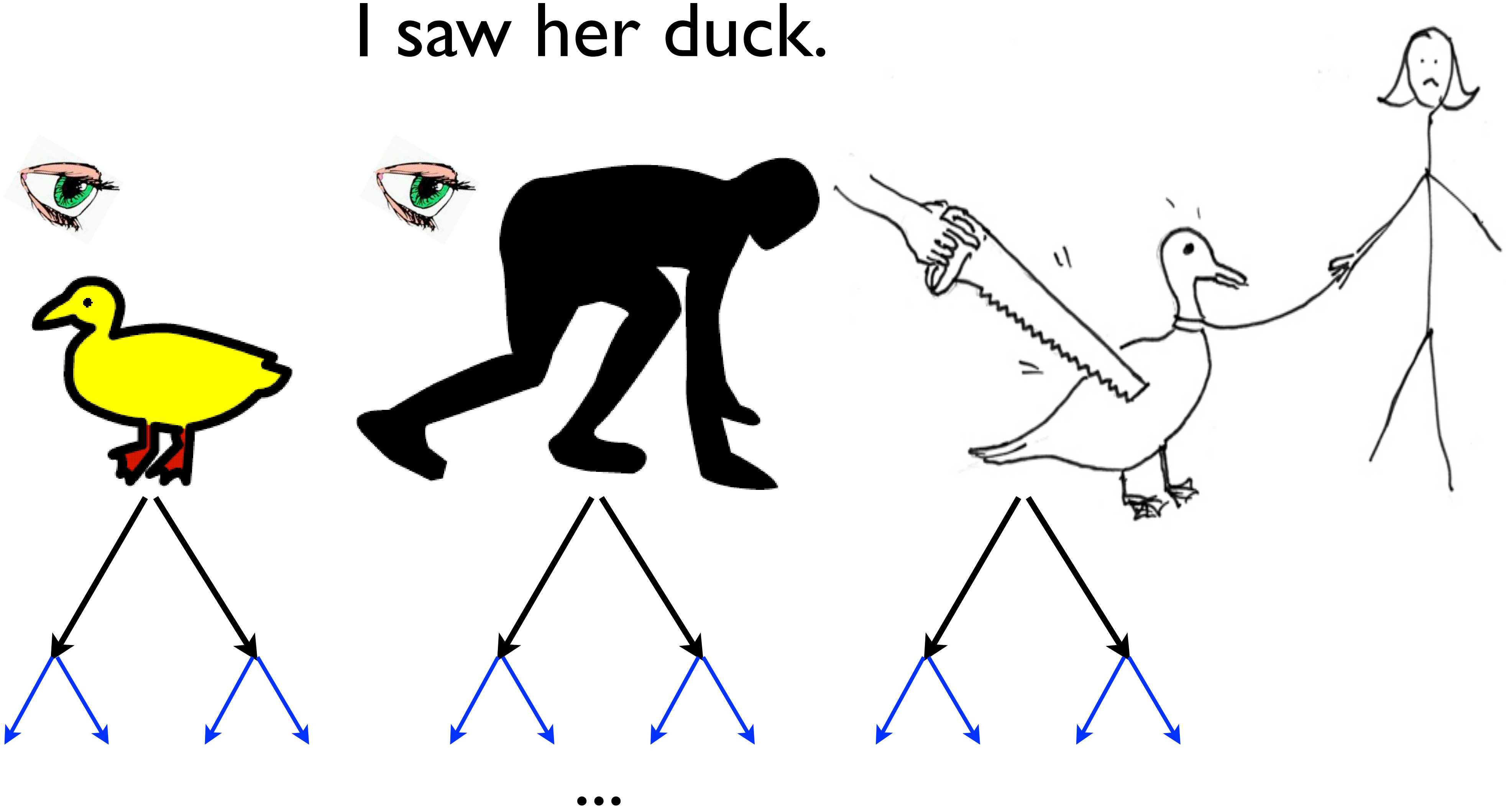
- Given the following CFG, how many parses exists for *the rose rose rose*?

- S→NP
- S →NP VP
- NP → DT NP2
- NP2 →JJ NP2
- NP2 →N N
- NP2 →N
- VP→V
- V →rose
- N → rose
- JJ →rose

slide by Liang Huang

Ambiguity Explosion by Recursion

I saw her duck.



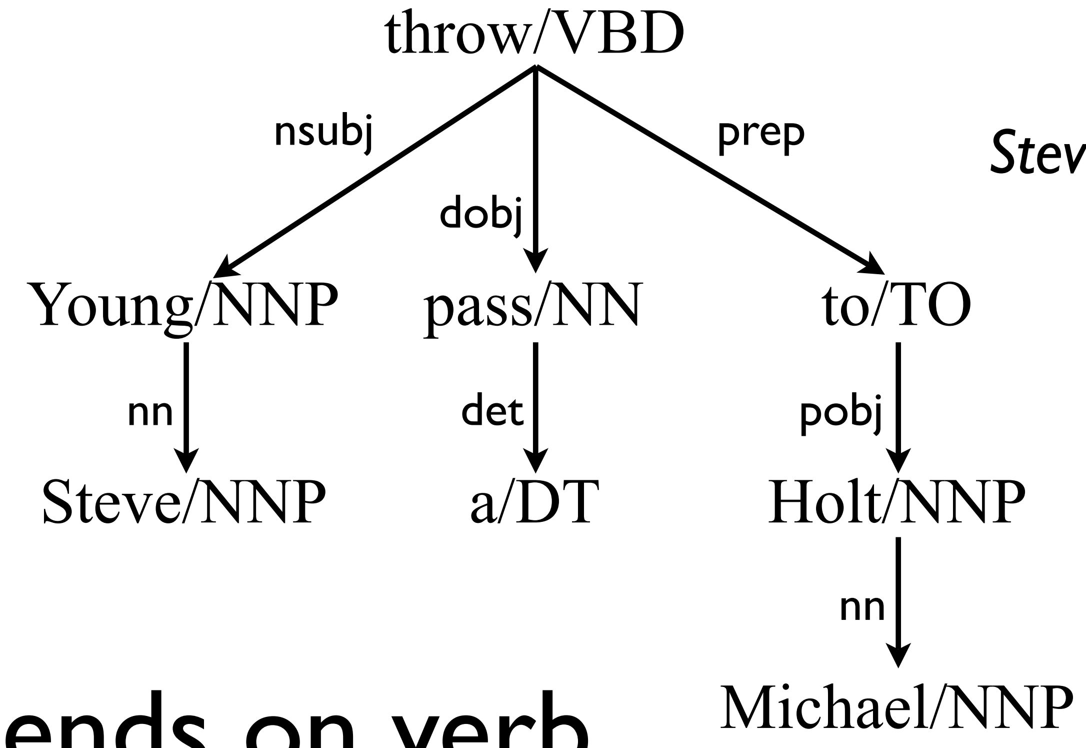
- how about...
  - I saw her duck with a telescope.
  - I saw her duck with a telescope in the garden...

Why do we care?

- parsing first step for most NLP tasks (MT, IE, IR, etc.)
  - disambiguate
  - find certain structures (noun phrases = chunking)
  - find syntactically related words

Other parsing

- dependency parsing: instead of constituents, find grammatical relations

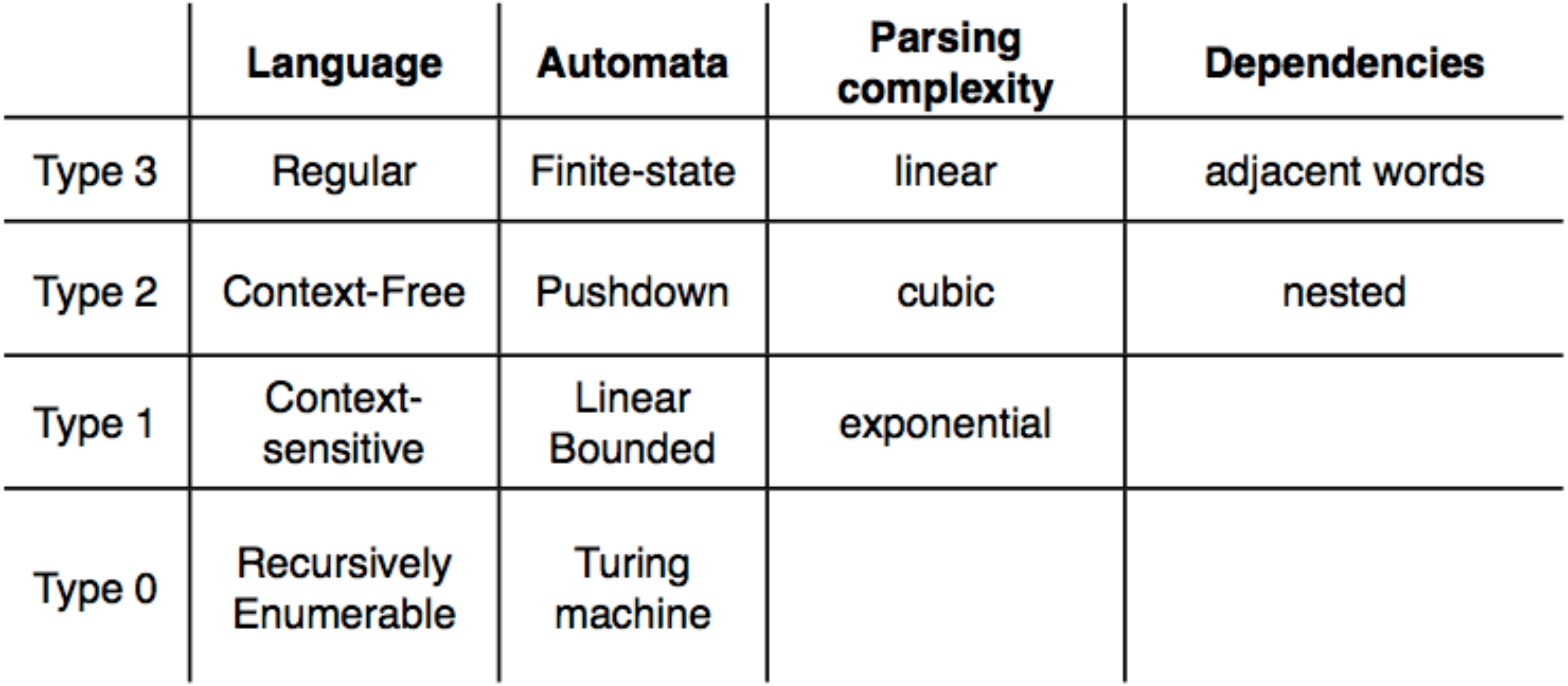


Steve Young threw a pass to Michael Holt

- depends on verb
- adds information
- less readable

slide by Liang Huang

Chomsky Hierarchy



	Language	Automata	Parsing complexity	Dependencies
Type 3	Regular	Finite-state	linear	adjacent words
Type 2	Context-Free	Pushdown	cubic	nested
Type 1	Context-sensitive	Linear Bounded	exponential	
Type 0	Recursively Enumerable	Turing machine		

computer science and linguistics share the same mathematical foundations.

In sum: Syntax

In sum: Syntax

- syntax = study of word order

I fart [in your general direction]
[on Sundays]
[with pleasure]

In sum: Syntax

- syntax = study of word order
- sentences consist of phrases (constituents)

I fart [in your general direction]
[on Sundays]
[with pleasure]

In sum: Syntax

- syntax = study of word order
- sentences consist of phrases (constituents)
- substitution can determine constituents

S → NP VP

In sum: Syntax

- syntax = study of word order
- sentences consist of phrases (constituents)
- substitution can determine constituents
- CFGs capture syntax rules

S → NP VP
NP → DT N

In sum: Syntax

- syntax = study of word order
- sentences consist of phrases (constituents)
- substitution can determine constituents
- CFGs capture syntax rules
- syntax rules are recursive

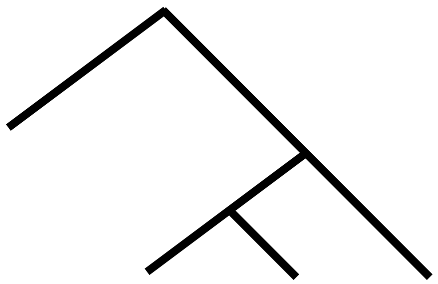
In sum: Syntax

- syntax = study of word order
- sentences consist of phrases (constituents)
- substitution can determine constituents
- CFGs capture syntax rules
- syntax rules are recursive

In sum: Parsing

In sum: Parsing

- parsers find rule structure of sentence

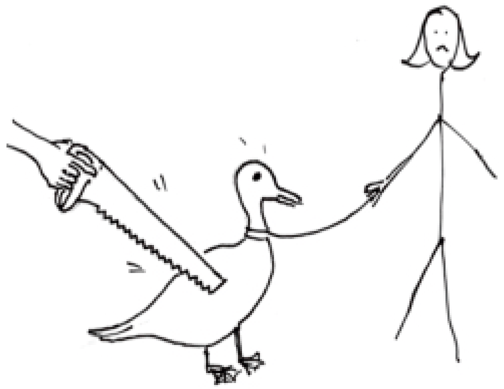


In sum: Parsing

- parsers find rule structure of sentence
- different strategies for search

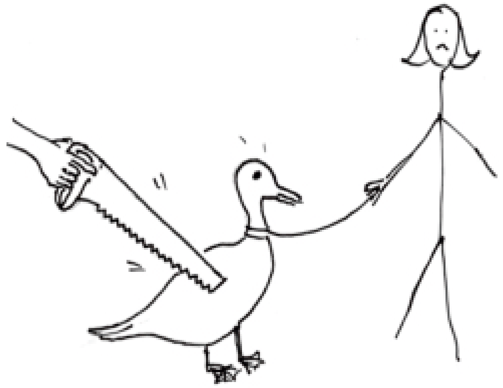
In sum: Parsing

- parsers find rule structure of sentence
- different strategies for search
- path b/w root and terminals



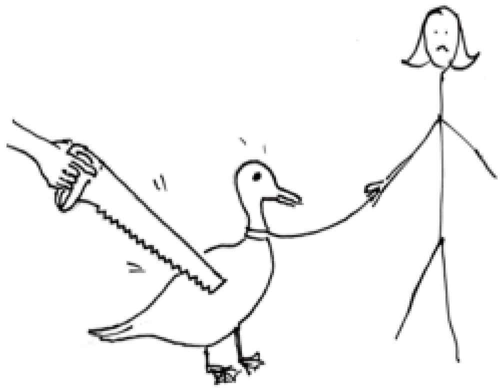
In sum: Parsing

- parsers find rule structure of sentence
- different strategies for search
- path b/w root and terminals
- language is ambiguous



In sum: Parsing

- parsers find rule structure of sentence
- different strategies for search
- path b/w root and terminals
- language is ambiguous
- parse trees are unambiguous



In sum: Parsing

- parsers find rule structure of sentence
- different strategies for search
- path b/w root and terminals
- language is ambiguous
- parse trees are unambiguous
- used to find structure, constituents, disambiguate words

If you learned nothing else:

- S → NP VP
- parsing is search

ask now or enjoy your afternoon…