SYNTAX

Matt Post
IntroHLT class
5 September 2024

Johns Hopkins University

So gorgeous was the spectacle on the May morning of 1910 when nine kings rode in the funeral of Edward VII of England that the crowd, waiting in hushed and black-clad awe, could not keep back gasps of admiration.

Barbara W. Tuchman, The Guns of August

morning keep could awe, the crowd, admiration. in hushed and black-clad of funeral May gorgeous of not on of rode waiting the VII England 1910 back that spectacle the Edward the in gasps kings was when nine of So

Other examples

GOOD		***BAD***
for i in range(args.N): print(i)	*python program*	i in: i for range(print)
<html> Lorem ipsum </html>	*HTML*	ipsum </p></html> <h> Lorem <ptml>
http://google.com	*URLs*	gsd@ht//:ww

What are the abstractions and tools that underlie all of these examples?

Today we will cover

math	**linguistics**	**engineering**
formal language theory	natural language	parsing
abstractions for reasoning about structure	*applying structure to natural phenomena*	*making them usable by a computer*

Goals for today

- After today, you should be able to
  - **describe syntax** both mathematically and linguistically
  - **enumerate** the formal language (Chomsky) hierarchy
  - **provide a description** of constituent grammars
  - **sketch the algorithm** for CKY parsing

Outline

formal language theory

natural language

parsing

Formal Language Theory

- Generalization: define a **language** to be a set of strings under some alphabet, Σ
  - e.g., the set of valid English sentences (where the "alphabet" is English words), or the set of valid Python programs

Formal Language Theory

- Generalization: define a **language** to be a set of strings under some alphabet, Σ
  - e.g., the set of valid English sentences (where the "alphabet" is English words), or the set of valid Python programs
- Formal Language Theory provides a common framework for studying properties of these languages, e.g.,
  - Is this file a valid C++ program? A valid Czech sentence?
  - What is the structure? <=> How do I find its meaning?
  - How hard / time-consuming is it to answer these questions?

Languages as sets

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

- What do you think these languages describe (in words?)

$$\mathscr{L}_1 = \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ \ldots\}$$
$$\mathscr{L}_2 = \{-12.4, 0,\ 142,\ 142.1,\ 142.01,\ 142.001,\ \ldots\}$$

Definitions

formal name	think…	description	repr
letter	**token**	the fundamental unit under consideration (e.g., a word, or a UTF-8-encoded letter)	$a, b, \ldots$
alphabet	**vocabulary**	A set of **tokens**	Σ
word	**"string"**	a sequence of zero or more **tokens** in the **vocabulary**	$\alpha, \beta, \ldots$
language	**language**	a set of strings	$\mathscr{L}$

Some notes

- $\Sigma*$ is the set of all strings in a vocabulary, Σ
- One special string is the empty string, {} or ϵ
- A language $\mathscr{L}$ can be very large—even infinite!
  - In fact, most languages probably are
  - List a few

Generative descriptions of lang.

- A definition of languages as **sets** is not very useful
  - Why not?
- A better approach:
  - Develop a process that can describe how strings in a language
  - New membership criteria:
    - **IN**: can be generated by this process
    - **OUT**: cannot be generated by this process
-

Generative examples

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- You probably know one generative process already: **regular expressions**
- What do these languages describe (in words?)

$$\mathscr{L}_1 = \Sigma*$$
$$\mathscr{L}_2 = 0 \mid [1-9][0-9]*$$

Regular language examples

- How can we write the following languages?
  - All floating point numbers
  - Email addresses
- These are much more compact representations compared to their set notations!

Regular languages with rules

- The "regular expression" syntax is a shortcut representation
- We can describe the generative process more formally using a set of *rules* that are recursively applied

$$A \rightarrow Aa$$
$$A \rightarrow a$$

- Rules have two types of symbols:
  - *terminal* symbols (lowercase letter) are normal vocabulary items
  - *nonterminal* symbols (capital letters) are recursively replaced until there are no more of them

Previous examples as rules

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

Previous examples as rules

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- $\mathscr{L}_1 = \Sigma*$
  S → A
  A → 0A
  A → 1A
  A → 2A
  …
  A → 9A

Previous examples as rules

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- $\mathscr{L}_1 = \Sigma*$
  S → A
  A → 0A
  A → 1A
  A → 2A
  …
  A → 9A
- A → ϵ

Previous examples as rules

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- $\mathscr{L}_2 = 0 \mid [1-9][0-9]*$

Previous examples as rules

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- $\mathscr{L}_2 = 0 \mid [1-9][0-9]*$
- S→A
  A→0
  A→1B
  A→2B
  …
  A→9B
  B→0B
  B→1B
  …
  B→9B
  B→ϵ

Formal definition of a language

- Definitions: consider the set $(\Sigma, N, S \in N, R)$, where
  - Σ is the *vocabulary* which is a finite set of *terminal symbols*
  - N is a finite set of *nonterminals symbols*
  - $S \in N$ is a special nonterminal called the *start symbol*
  - $\alpha, \beta,$ *and* γ are *strings* of zero or more terminal and nonterminal symbols
  - R is a set of *rules* of the form $\alpha N \beta \rightarrow \gamma$

Regular languages

- Definitions: consider the set $(\Sigma, N, S \in N, R)$, where
  - Σ is the *vocabulary* which is a finite set of *terminal symbols*
  - N is a finite set of *nonterminals symbols*
  - $S \in N$ is a special nonterminal called the *start symbol*
  - $\alpha, \beta,$ *and* γ are *strings* of zero or more terminal and nonterminal symbols
  - R is a set of *rules* of the form $\alpha N \beta \rightarrow \gamma$

Type	Rules	Name	Recognized by
3	A → aB	Regular	Regular expressions

- All the languages we created earlier (for example, the set of email addresses) can be described with such rules

Context-free languages

- Definitions: consider the set $(\Sigma, N, S \in N, R)$, where
  - Σ is the *vocabulary* which is a finite set of *terminal symbols*
  - N is a finite set of *nonterminals symbols*
  - $S \in N$ is a special nonterminal called the *start symbol*
  - $\alpha, \beta,$ *and* γ are *strings* of zero or more terminal and nonterminal symbols
  - R is a set of *rules* of the form $\alpha N \beta \rightarrow \gamma$

Type	Rules	Name	Recognized by
2	$A \rightarrow \alpha$	Context-free	Pushdown automata

- This change might seem small, but it fundamentally alters the kinds of languages that can be generated

Context-free and not regular

- $\Sigma = \{a, b, c, \ldots, z\}$
- Create a context-free language for $\mathscr{L}$, the set of palindromes

Context-free and not regular

- $\Sigma = \{a, b, c, \ldots, z\}$
- Create a context-free language for $\mathscr{L}$, the set of palindromes
- S→A
  A→aAa
  A→bAb
  …
  A→zAz
  A→ϵ

Context-free and not regular

- $\Sigma = \{a, b, c, \ldots, z\}$
- Create a context-free language for $\mathscr{L}$, the set of palindromes
- S→A
  A→aAa
  A→bAb
  …
  A→zAz
  A→ϵ
- Can you do this with the regular language constraint on rules?

The Chomsky Hierarchy

- Named after Noam Chomsky, the MIT linguist
- Different constraints on the rules lead to more powerful sets of languages that can be described
- More powerful languages are harder (meaning, more compute-intensive) to recognize

The Chomsky Hierarchy

- Definitions: consider the set $(\Sigma, N, S \in N, R)$, where
  - Σ is the *vocabulary* which is a finite set of *terminal symbols*
  - N is a finite set of *nonterminals symbols*
  - $S \in N$ is a special nonterminal called the *start symbol*
  - $\alpha, \beta,$ *and* γ are *strings* of zero or more terminal and nonterminal symbols
  - R is a set of *rules* of the form $\alpha N \beta \rightarrow \gamma$

Type	Rules	Name	Recognized by	Complexity
3	A → aB	Regular	Regular expressions	$\mathcal{O}(n)$
2	A → α	Context-free	Pushdown automata	$\mathcal{O}(n^3)$
1	αAβ → $\alpha\gamma\beta$	Context-sensitive	Linear-bounded Turing machine	$\mathcal{O}(2^n)$
0	αAβ → γ	Recursively enumerable	Turing Machines	undecidable

Summary

- This view of languages (not sets, but “capturing” generative processes) is very productive

Summary

- This view of languages (not sets, but “capturing” generative processes) is very productive
- We can generalize this discussion to make a connection between natural and other kinds of languages

Summary

- This view of languages (not sets, but “capturing” generative processes) is very productive
- We can generalize this discussion to make a connection between natural and other kinds of languages
- Consider, for example, *computer programs*
  - They either compile or don’t compile
  - *Their structure determines their interpretation*

Summary

- This view of languages (not sets, but “capturing” generative processes) is very productive
- We can generalize this discussion to make a connection between natural and other kinds of languages
- Consider, for example, *computer programs*
  - They either compile or don’t compile
  - *Their structure determines their interpretation*
- What is the structure?

Outline

formal language theory

natural language

parsing

Linguistic fields of study

- Phonetics: sounds

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure
- **Syntax**: external word structure (sentences)

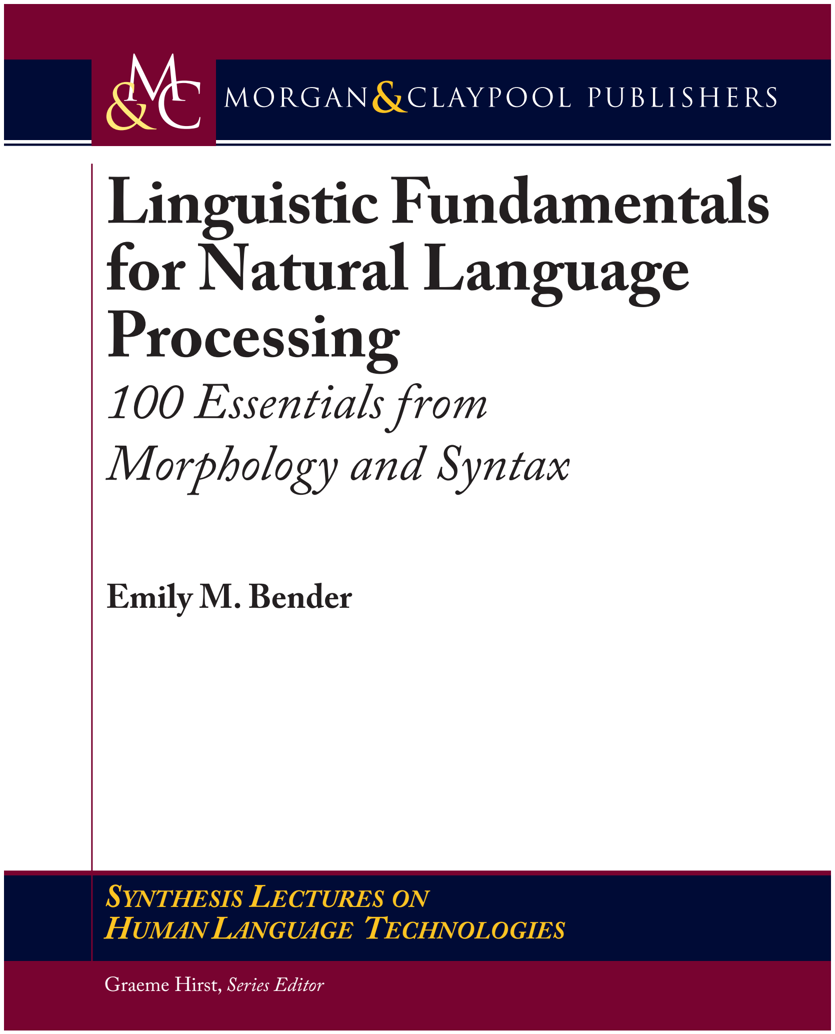
Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure
- **Syntax**: external word structure (sentences)
- Semantics: sentence meaning

Linguistic fields of study

- Phonetics: sounds
- Phonology: sound systems
- Morphology: internal word structure
- **Syntax**: external word structure (sentences)
- Semantics: sentence meaning
- Pragmatics: contextualized meaning and communicative goals

Today's focus



- Excellent book
- Organized into 100 mini-lectures
- PDF available for free via JHU library (along with tens of others in the series)
- https://tinyurl.com/linguistic-fundamentals

What is syntax?

- A set of constraints on the possible sentences in the language
  - *A set of <u>constraint</u> on the possible <u>sentence.</u>
  - *Dipanjan had <u>[a]</u> question.
  - *You are <u>on</u> class.
- At a coarse level, we can divide all possible sequences of words into two groups: *valid* and *invalid* (or *grammatical* and *ungrammatical*)

POS Examples

- No general agreement about the exact set of parts of speech
- One set of examples from the Penn Treebank

POS Examples

- No general agreement about the exact set of parts of speech
- One set of examples from the Penn Treebank
  - nouns: NN, NNS, NNP, NNPS

POS Examples

- No general agreement about the exact set of parts of speech
- One set of examples from the Penn Treebank
  - nouns: NN, NNS, NNP, NNPS
  - adverbs: RB, RBR, RBS, RP

POS Examples

- No general agreement about the exact set of parts of speech
- One set of examples from the Penn Treebank
  - nouns: NN, NNS, NNP, NNPS
  - adverbs: RB, RBR, RBS, RP
  - verbs: VB, VBD, VBG, VBN, VBP, VBZ

POS Examples

- No general agreement about the exact set of parts of speech
- One set of examples from the Penn Treebank
  - nouns: NN, NNS, NNP, NNPS
  - adverbs: RB, RBR, RBS, RP
  - verbs: VB, VBD, VBG, VBN, VBP, VBZ
  - (Here, different tags are used to capture the small bit of morphology present in English)

Parts of Speech (POS)

- Three definitions of **noun**

Grammar school
("metaphysical")
a person, place,
thing, or idea

Parts of Speech (POS)

- Three definitions of **noun**

<u>Grammar school</u>	<u>Distributional</u>
(“metaphysical”)	
a person, place, thing, or idea	*the set of words that have the same distribution as other nouns*
	{I,you,he} saw the {bird,cat,dog}.

Parts of Speech (POS)

- Three definitions of **noun**

<u>Grammar school</u> (“metaphysical”)

a person, place, thing, or idea

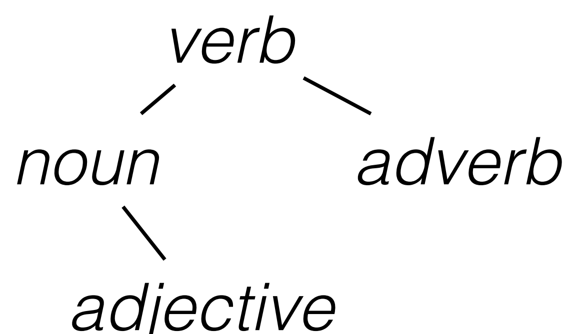
<u>Distributional</u>

the set of words that have the same distribution as other nouns

{I,you,he} saw the {bird,cat,dog}.

<u>Functional</u>

the set of words that serve as arguments to verbs



Phrases and Constituents

- Longer sequences of words can perform the same function as individual parts of speech:
  - I saw [a_{DT} kid_{N}]$_{NP}$
  - I saw [a kid playing basketball]$_{NP}$
  - I saw [a kid playing basketball alone on the court]$_{NP}$
- This gives rise to the idea of a *phrasal constituent*, which functions as a unit in relation to the rest of the sentence

Constituent tests

- How do you know if a phrase functions as a constituent?
- A few tests
  - *Coordination*
    - Kim [read a book], [gave it to Sandy], and [left].
  - *Substitution with a word*
    - Kim read [a very interesting book about grammar].
    - Kim read [it].
  - See Bender #51

Constituent structure

- The head often constrains the internal structure of a constituent
- Examples
  - verb
    - [Kim]^{ARGUMENT} **is** [ready]^{ADJUNCT}.
  - adjective
    - Kim is [**ready**$_{ADJ}$ [to make a pizza]$_{V}$].
    - * Kim is [**tired**$_{ADJ}$ [to make a pizza]$_{V}$].
  - noun
    - [The [red]$_{ADJ}$ **ball**]
    - * [The [red]$_{ADJ}$ **ball** [the stick]$_{N}$]
    - [The [red]$_{ADJ}$ **ball** [on top of the stick]$_{PP}$]

More examples

- Kim **planned** [to **give** Sandy books].
- * Kim **planned** [to **give** Sandy].
- Kim **planned** [to give books].
- * Kim **planned** [to **see** Sandy books].
- Kim [**would** [**give** Sandy books]].
- Pat [**helped** [Kim **give** Sandy books]].
- * [[**Give** Sandy books] [**surprised** Kim]].

Human judgments

- How do we know what's in and out? We simply ask humans
- But how do humans know? This is the tie-in to formal language theory

Context Free Grammar

- A *finite set of* ***rules*** licensing a (possibly infinite) *number of* ***strings***

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of* ***rules*** licensing a (possibly infinite) *number of* ***strings***
- e.g., some rules

Chomsky formal language hierarchy refresher

finite state machine

context-free grammar

context-sensitive grammar

Turing machine

Context Free Grammar

- A *finite set of **rules*** licensing a (possibly infinite) *number of **strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of **rules*** licensing a (possibly infinite) *number of **strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of* ***rules*** licensing a (possibly infinite) *number of* ***strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of **rules*** licensing a (possibly infinite) *number of **strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]
  - [predicate] → [verb phrase] [adjunct]

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of **rules*** licensing a (possibly infinite) *number of **strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]
  - [predicate] → [verb phrase] [adjunct]
- Rules are *phrasal* or *terminal*

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of **rules*** licensing a (possibly infinite) *number of **strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]
  - [predicate] → [verb phrase] [adjunct]
- Rules are *phrasal* or *terminal*
  - Phrasal rules form **constituents** in a tree

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Context Free Grammar

- A *finite set of **rules*** licensing a (possibly infinite) *number of **strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]
  - [predicate] → [verb phrase] [adjunct]
- Rules are *phrasal* or *terminal*
  - Phrasal rules form **constituents** in a tree
  - Terminal rules are **parts of speech** and produce words

Chomsky formal language hierarchy refresher

finite state machine
context-free grammar
context-sensitive grammar
Turing machine

Example

Example

S → NP VP .	S → NP VP

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS

Example

S → NP VP .
S → [JJ NNS] VP .
S → [Human] NNS VP .

S → NP VP
VP → JJ NNS
JJ → Human

Example

S → NP VP .
S → [JJ NNS] VP .
S → [Human] NNS VP .
S → Human [languages] VP .

S → NP VP
VP → JJ NNS
JJ → Human
NNS → languages

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard
S → Human languages are hard [VP] .	SBAR → S, S → VP

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard
S → Human languages are hard [VP] .	SBAR → S, S → VP
S → Human languages are hard [TO VP] .	VP → TO VP

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard
S → Human languages are hard [VP] .	SBAR → S, S → VP
S → Human languages are hard [TO VP] .	VP → TO VP
S → Human languages are hard [to] VP .	TO → to

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard
S → Human languages are hard [VP] .	SBAR → S, S → VP
S → Human languages are hard [TO VP] .	VP → TO VP
S → Human languages are hard [to] VP .	TO → to
S → Human languages are hard to [VB] .	VP → VB

Example

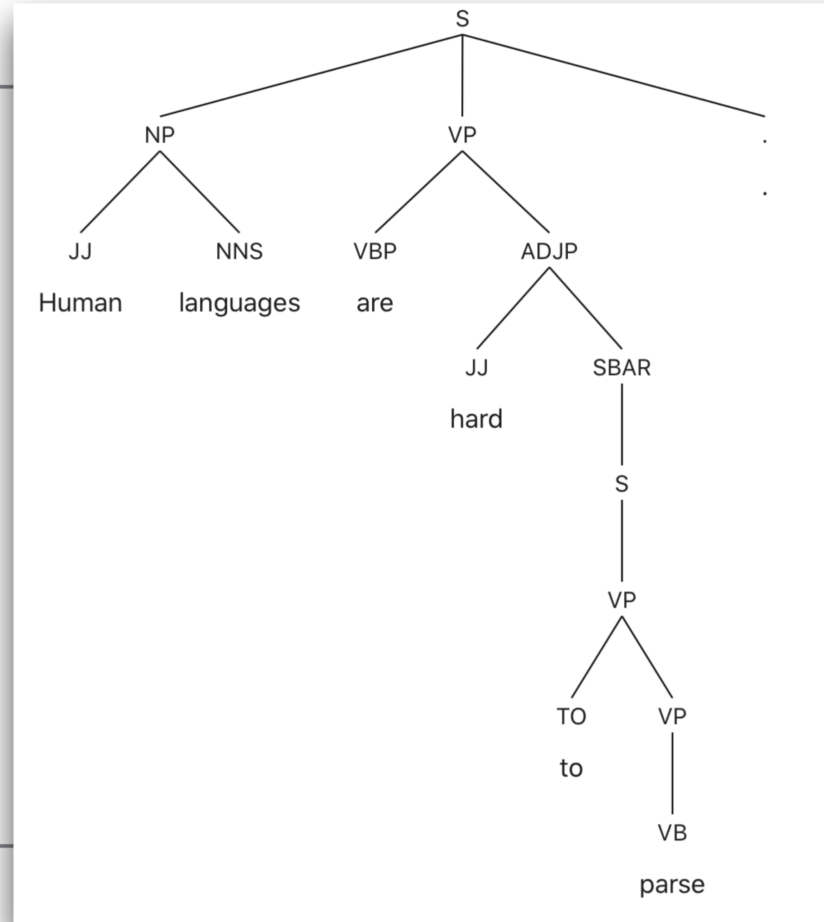
S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard
S → Human languages are hard [VP] .	SBAR → S, S → VP
S → Human languages are hard [TO VP] .	VP → TO VP
S → Human languages are hard [to] VP .	TO → to
S → Human languages are hard to [VB] .	VP → VB
S → Human languages are hard to [parse] .	VB → parse

Example

S → NP VP .	S → NP VP
S → [JJ NNS] VP .	VP → JJ NNS
S → [Human] NNS VP .	JJ → Human
S → Human [languages] VP .	NNS → languages
S → Human languages [VBP ADJP] .	VP → VBP ADJP
S → Human languages [are] ADJP .	VBP → are
S → Human languages are [JJ SBAR] .	ADJP → JJ SBAR
S → Human languages are [hard] SBAR .	JJ → hard
S → Human languages are hard [VP] .	SBAR → S, S → VP
S → Human languages are hard [TO VP] .	VP → TO VP
S → Human languages are hard [to] VP .	TO → to
S → Human languages are hard to [VB] .	VP → VB
S → Human languages are hard to [parse] .	VB → parse
S → Human languages are hard to parse .	■

Example

S → NP VP .
S → [JJ NNS] VP .
S → [Human] NNS VP .
S → Human [languages] VP .
S → Human languages [VBP ADJP] .
S → Human languages [are] ADJP .
S → Human languages are [JJ SBAR] .
S → Human languages are [hard] SBAR .
S → Human languages are hard [VP] .
S → Human languages are hard [TO VP] .
S → Human languages are hard [to] VP .
S → Human languages are hard to [VB] .
S → Human languages are hard to [parse] .
S → Human languages are hard to parse .



Treebanks

- Collections of natural text that are annotated according to a particular syntactic theory
  - Usually created by linguistic experts
  - Ideally as large as possible
  - Theories are usually coarsely divided into *constituent/ phrase* or *dependency* structure

Penn Treebank (1993)

https://catalog.ldc.upen

Login or Register

- ABOUT
- MEMBERS
- COMMUNICATIONS
- LANGUAGE RESOURCES
  - Data
    - Obtaining Data
    - Catalog
    - By Year
    - Top Ten Corpora
    - Projects
    - Search
    - Memberships
    - Data Scholarships
  - Tools
  - Papers
  - LR Wiki
- DATA MANAGEMENT
- COLLABORATIONS

Home › Language Resources › Data

Treebank-3

Item Name:	Treebank-3
Author(s):	Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, Ann Taylor
LDC Catalog No.:	LDC99T42
ISBN:	1-58563-163-9
ISLRN:	141-282-691-413-2
Member Year(s):	1999
DCMI Type(s):	Text
Data Source(s):	telephone speech, newswire, microphone speech, transcribed speech, varied
Project(s):	TIDES, GALE
Application(s):	parsing, natural language processing, tagging
Language(s):	English
Language ID(s):	eng
License(s):	LDC User Agreement for Non-Members
Online Documentation:	LDC99T42 Documents
Licensing Instructions:	Subscription & Standard Members, and Non-Members
Citation:	Marcus, Mitchell, et al. Treebank-3 LDC99T42. Web Download. Philadelphia: Linguistic Data Consortium, 1999.
Related Works:	View

Introduction

This release contains the following Treebank-2 Material:

- One million words of 1989 Wall Street Journal material annotated in Treebank II style.
- A small sample of ATIS-3 material annotated in Treebank II style.
- A fully tagged version of the Brown Corpus.

and the following new material:

- Switchboard tagged, dysfluency-annotated, and parsed text
- Brown parsed text

The Treebank bracketing style is designed to allow the extraction of simple predicate/argument structure. Over one million words of text are provided with this bracketing applied.

Data

The Penn Treebank

- Syntactic annotation of a million words of the 1989 Wall Street Journal, plus other corpora (released in 1993)
  - (Trivia: People often discuss “The Penn Treebank” when the mean the WSJ portion of it)

The Penn Treebank

- Syntactic annotation of a million words of the 1989 Wall Street Journal, plus other corpora (released in 1993)
  - (Trivia: People often discuss “The Penn Treebank” when the mean the WSJ portion of it)
- Contains 74 total tags: 36 parts of speech, 7 punctuation tags, and 31 phrasal constituent tags, plus some relation markings

The Penn Treebank

- Syntactic annotation of a million words of the 1989 Wall Street Journal, plus other corpora (released in 1993)
  - (Trivia: People often discuss “The Penn Treebank” when the mean the WSJ portion of it)
- Contains 74 total tags: 36 parts of speech, 7 punctuation tags, and 31 phrasal constituent tags, plus some relation markings
- Was the foundation for an entire field of research and applications for over twenty years

```
( (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) )))
    (. .) ))
```

https://commons.wikimedia.org/wiki/File:PierreVinken.jpg

Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD        S years) )
        (JJ old)
      (, ,) )
    (VP (MD wil )
      (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) )))
    (. .) ))
```

x 49,208

https://commons.wikimedia.org/wiki/File:PierreVinken.jpg

Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

Summary

- Formal language theory is a theory that does the following:
  - provides a compact representation of a language
  - provides an account for how strings within a language are generated
- It's very useful for describing many simple languages
- It can also be applied to natural language

Outline

formal language theory

natural language

parsing

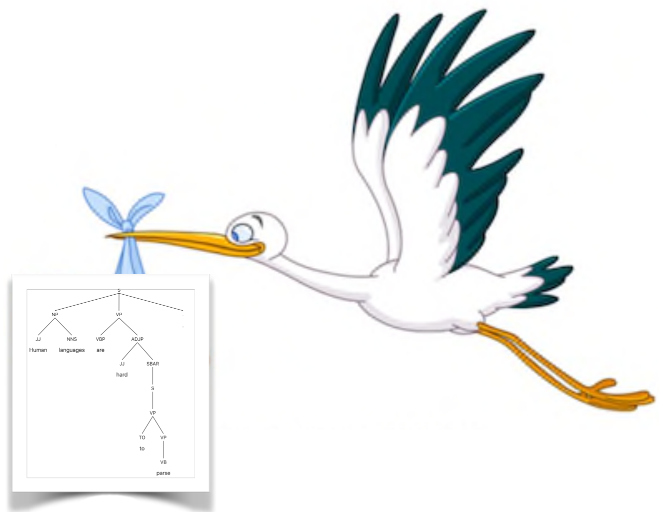
Where we are

- We discussed formal language theory
- We showed how it might apply to human language
- But how do we get a computer to use it?
  - Sentences (or other strings we wish to parse) are observed; the structure is hidden
  - We assume these were generated by a model
  - We need
    - An algorithm for finding the sequence of actions under that model, most likely to have produced it
    - A way to learn that model

Where do grammars come from?

Where do grammars come from?

- Treebanks!
- Given a treebank, and a formalism, we can learn statistics by counting over the annotated instances



I stole this joke from Chris Callison-Burch

https://www.shutterstock.com/image-vector/stork-carrying-baby-boy-133823486

Probabilities

- For example, a context-free grammar
- We can get probabilities by reading all instances from a Treebank

$$P(A \rightarrow B\ C) = \sum_{A' \in N} \frac{P(A)}{P(A')}$$

← *a CFG rule*

← *all CFG rules with the same lefthand side*

- e.g.,
  - S → NP , NP VP . [0.002]
  - NP → NNP NNP [0.037]
  - , → , [0.999]
  - NP → * [X]
  - VP → VB NP [0.057]
  - NP → PRP$ NN [0.008]
  - . → . [0.987]

Parsing

- If the grammar has certain properties (Type 2 or 3), we can efficiently answer the first question (find the hidden structure) with a **parser**
  - **Q1**: is the sentence in the language of the parser?
  - **Q2**: What is the structure above that sentence?

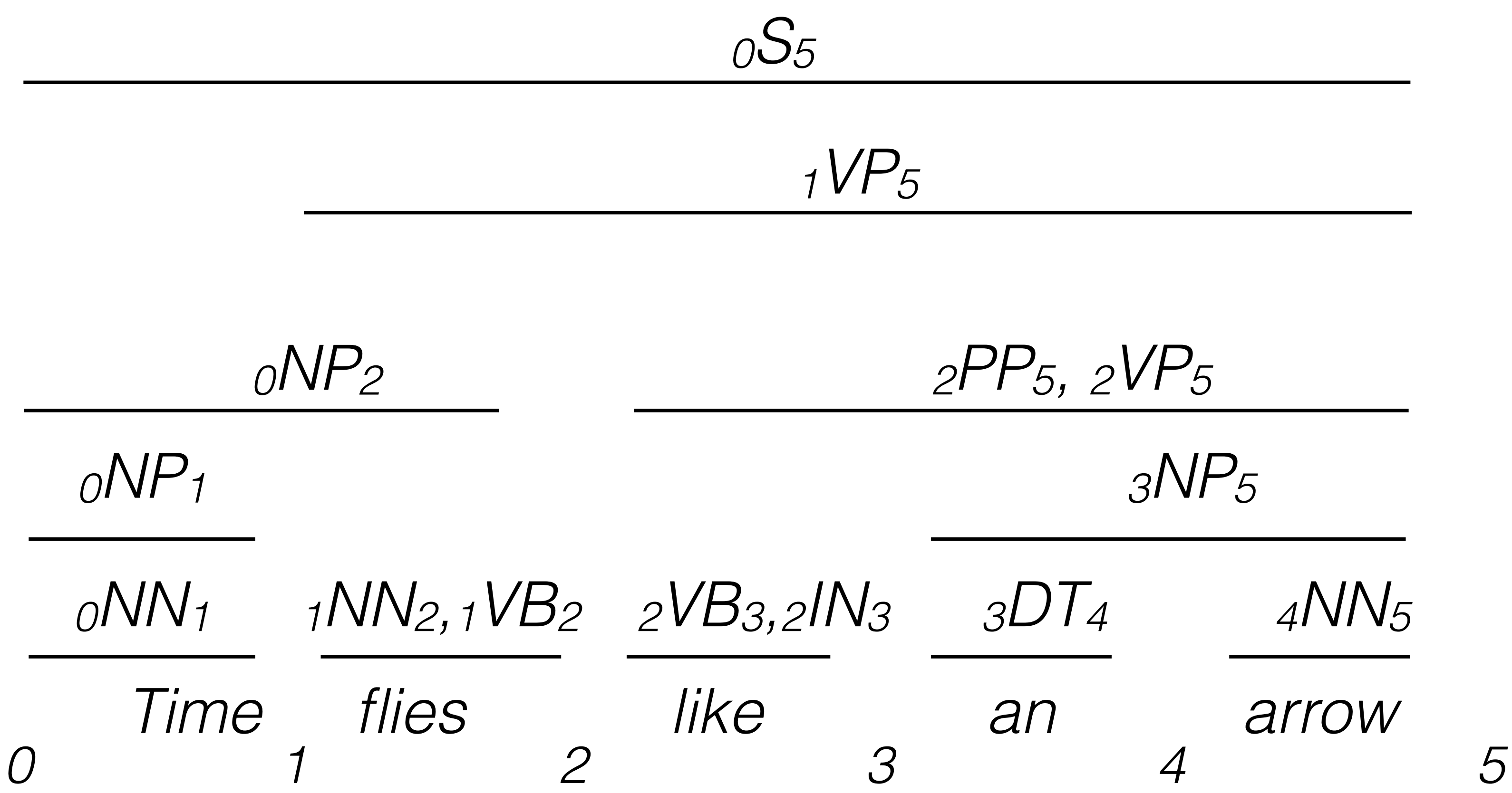
Algorithms

- The **CKY algorithm** for parsing with constituency grammars

Chart parsing for constituency grammars

- Maintains a chart of nonterminals spanning words, e.g.,
  - NP over words 1..4 and 2..5
  - VP over words 4..6 and 4..8
  - etc
- Build this chart from the bottom upward: the *opposite* direction from generation

Chart parsing for constituency grammars



CKY algorithm

- How do we produce this chart? Cocke-Younger-Kasami (CYK/ CKY)
- Basic idea is to apply rules in a bottom-up fashion, applying all rules, and (recursively) building larger constituents from smaller ones
- Input: sentence of length N

for width in 2..N

  for begin **i** in 1..{N - width}

  j = i + width

    for split **k** in {i + 1}..{j - 1}

      for all rules A → B C

        create $_i\mathbf{A}_j$ if $_i\mathbf{B}_k$ and $_k\mathbf{C}_j$

CKY algorithm

Time flies like an arrow

0 1 2 3 4 5

CKY algorithm

NN	*NN,VB*	*VB,IN*	*DT*	*NN*
Time	*flies*	*like*	*an*	*arrow*

0 *1* *2* *3* *4* *5*

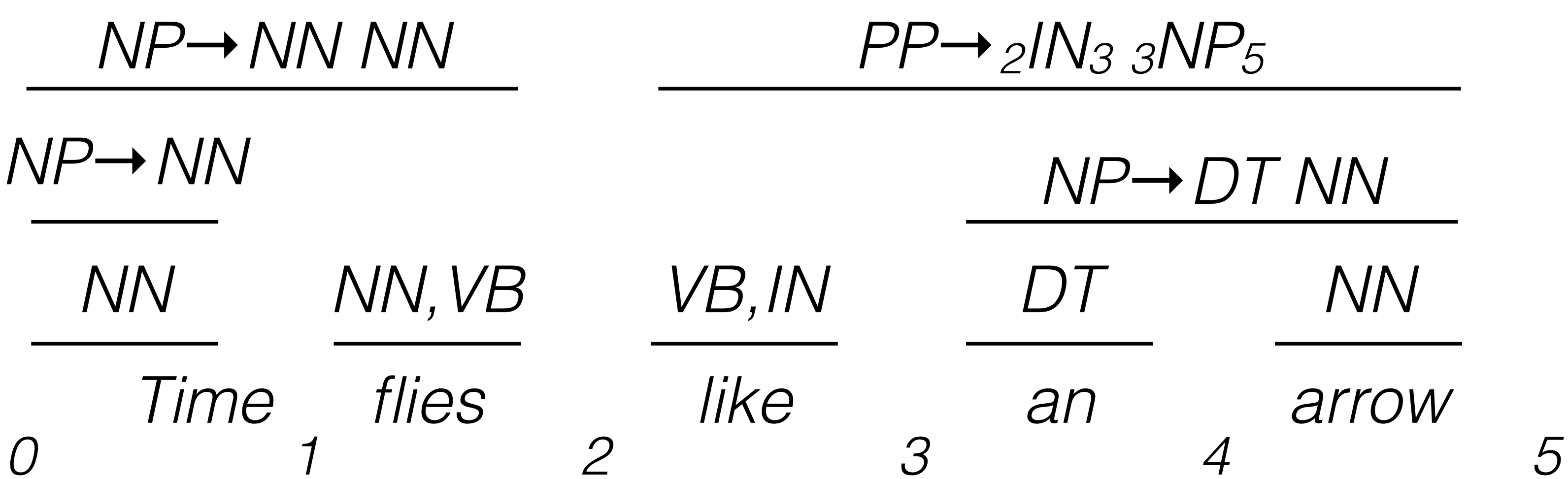
CKY algorithm

NP→NN   NP→DT NN

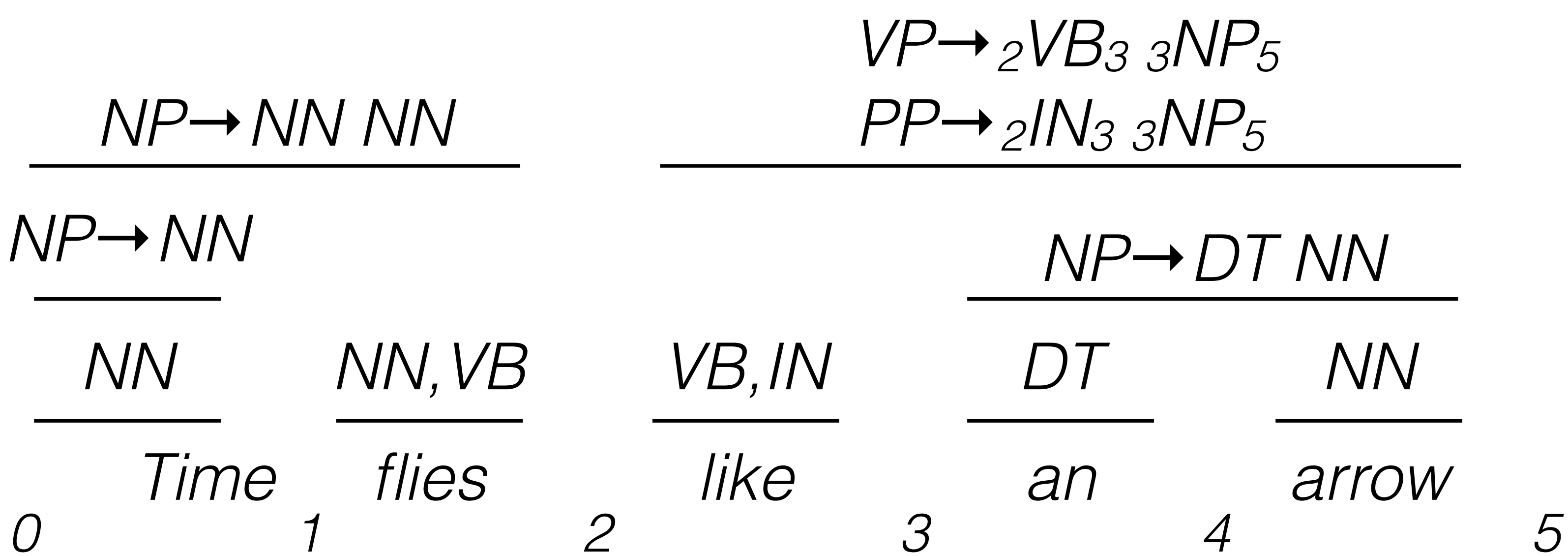
NN	NN,VB	VB,IN	DT	NN
Time	flies	like	an	arrow

0   1   2   3   4   5

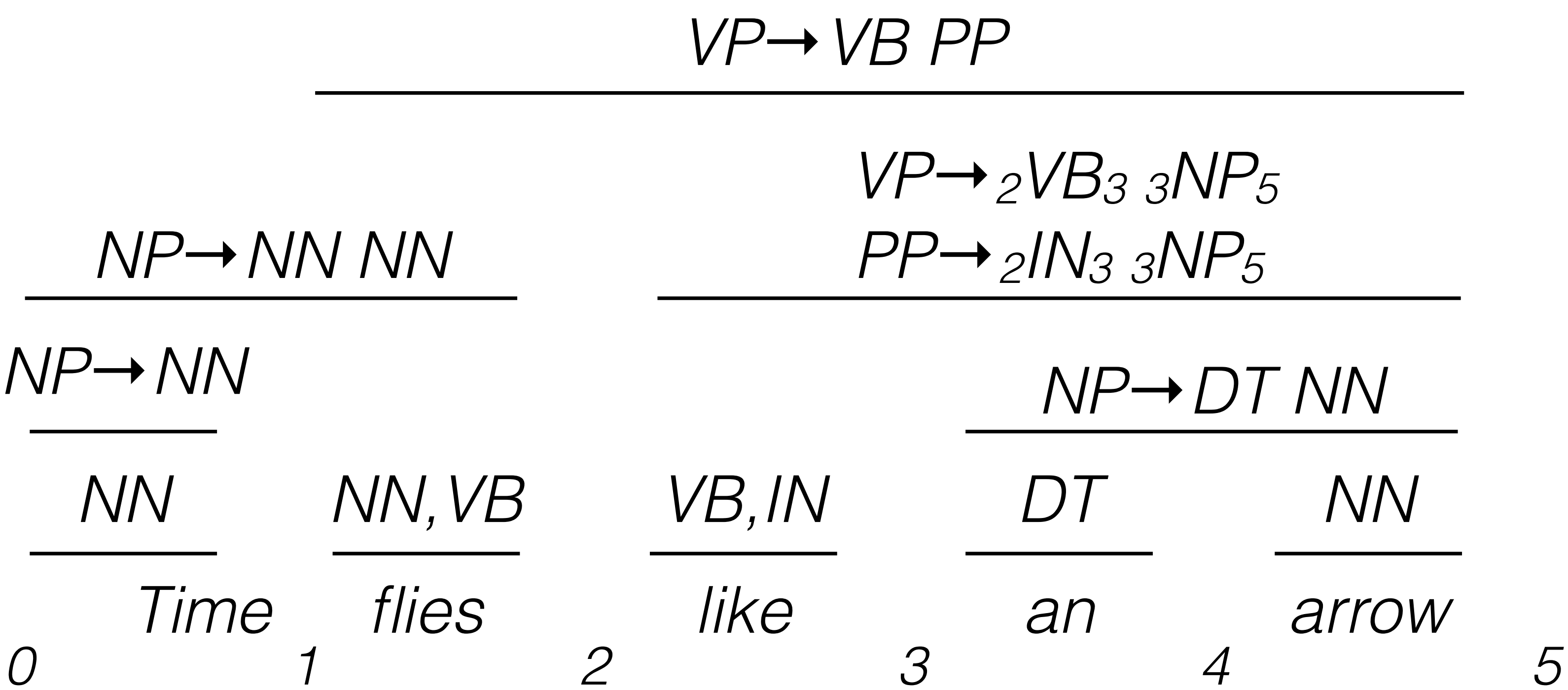
CKY algorithm



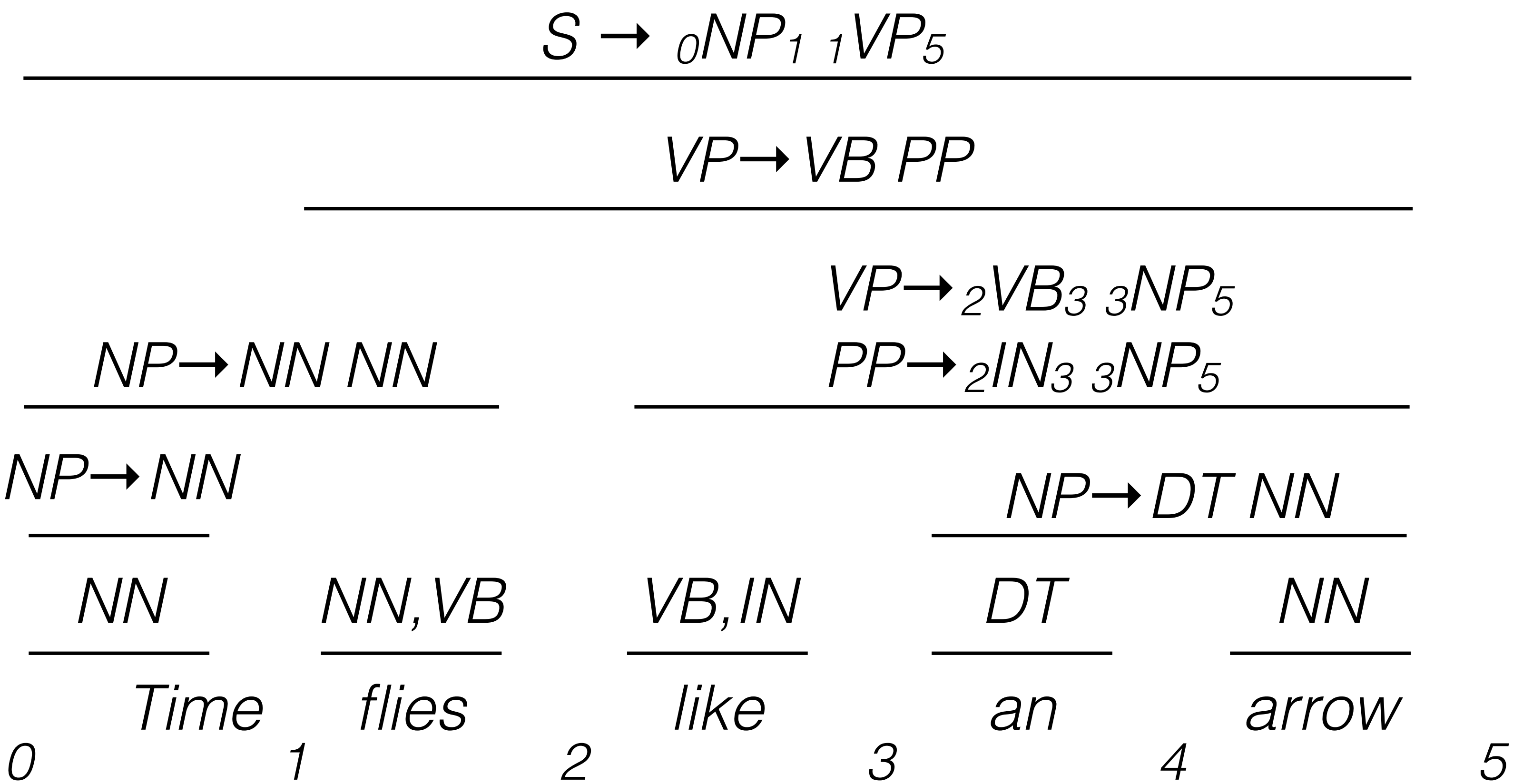
CKY algorithm



CKY algorithm



CKY algorithm



CKY algorithm

$S \rightarrow {}_0NP_2\ {}_2VP_5$

$S \rightarrow {}_0NP_1\ {}_1VP_5$

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN	NN,VB	VB,IN	DT	NN
Time	flies	like	an	arrow

0 1 2 3 4 5

CKY algorithm

- Parsing questions:
  - **Q1**: is a given sentence in the language of the parser?
  - **Q2**: What is the structure above that sentence?
- Termination: is there a chart entry at ${}_0S_N$?
  - ✓ string is in the language (Q1)
  - Structures can be obtained by following backpointers in dynamic programming chart (not covered today)
- Other technical details not covered today:
  - The probability of each parse is the product of the rule probabilities
  - Ambiguities are resolved with these scores

- Demos:
  - Berkeley Neural Parser: https://parser.kitaev.io
  - Spacy dependency parser: https://explosion.ai/demos/displacy

Summary

formal language theory	natural language	parsing
provides a framework for reasoning about languages of all kinds	*a real-world (if messy) application area for formal language theory*	*a means of making text useable under formal language theory*