SYNTAX

Matt Post
IntroHLT class
11 September 2025

Johns Hopkins University

Languages

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
The crowd could not keep back gasps of admiration	*language*	not of could back gasps The crowd admiration

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

Today we will cover

math

formal language theory

abstractions for reasoning about structure

linguistics

natural language

applying structure to natural phenomena

engineering

parsing

making them usable by a computer

Goals for today

- After today, you should be able to
  - **define** a language
  - **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

- Define a **language** to be a set of strings under some alphabet, Σ
  - Σ = a set of symbols (letters, words, numbers, etc)
  - string = a sequence of symbols from 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

Some terminology

- Σ^* ("sigma star") is the set of all strings in the vocabulary
- ϵ is the *empty string*
- A language $\mathscr{L}$ can be finite or infinite

Languages as sets

- $\Sigma = \{0,1,2,3,4,5,6,7,8,9, - , . \}$
- What do you think these languages describe (in words?)

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

Regular expression examples

- A more compact representation than explicit listing: **regular expressions**
- Notes:
  - . = match any character
  - | = “or”
  - [] = “choose one of these”
  - + = “one or more of the previous”
  - * = “zero or more of the previous”
- What do these languages describe (in words?)

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

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 are generated
  - New membership criteria:
    - **IN**: can be generated by this process
    - **OUT**: cannot be generated by this process
-

Generating from a language

- Imagine a process that repeatedly
  - selects a next symbol from the alphabet under some strategy
  - decides whether to terminate
- How can we formalize this?

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, e.g., *a*) are normal vocabulary items
  - *nonterminal* symbols (uppercase, e.g., *A*) are recursively replaced until there are no more of them

Previous examples as rules

- Alphabet: $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- Language: $\mathscr{L}_1 = \Sigma^+$
- Generative rules: S → A
  A → 0A
  A → 1A
  A → 2A
  …
  A → 9A
- A → ϵ

How can we modify this to prevent the empty string?

Previous examples as rules

- Alphabet: $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
- Language: $\mathscr{L}_1 = \Sigma^+$
- Generative rules: S → A
  A → 0B
  A → 1B
  A → 2B
  …
  A → 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 $\alpha N \beta \to \gamma$

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

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

Palindromes

- "No sir, away, a papaya war is on!"
  - aibohphobia—the fear of palindromes
- "engage le jeu que je le gagne"
- Belphegor's prime: 1000000000000066600000000000001

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

Exercise

- Can you construct a grammar recognizing palindromes using the **regular** constraint on grammar rules?
  - Rules of the form A → Aa or A → aA
  - (Nonterminals must be on one side or the other)

Nonterminals as categories

- We can give meaning to the categories
- START→REPEAT
  REPEAT→a REPEAT a
  REPEAT→b REPEAT b
  …
  REPEAT→z REPEAT z
  REPEAT→ϵ

  REPEAT → a DONE a
  REPEAT → b DONE b
  ...
  REPEAT → z DONE z
  DONE → ϵ

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

- Given all strings under a vocabulary, a language can be thought of as a subset of those strings
- It is productive to formulate languages as the set of strings produced by a generative process
- We can generalize this discussion to make a connection between natural and other kinds of languages
- Consider, for example, *computer programs*, where the set of Python programs is the subset of strings that can be parsed by the Python interpreter

Outline

formal language theory

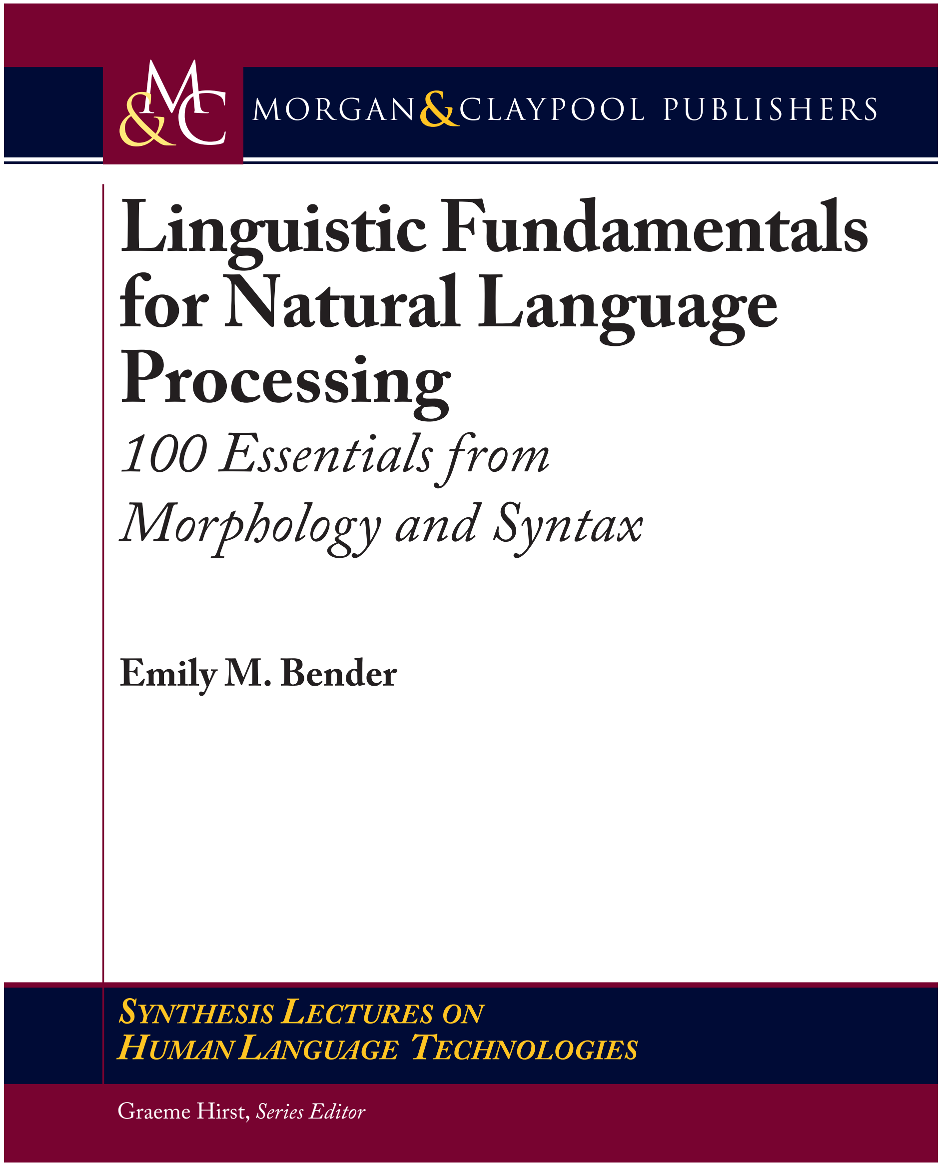
natural language

parsing

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

Human judgments

- “Ungrammatical” = “not in the language”
- Proficient speakers make these judgments
- But we still want to try to model the process

Parts of speech

- No general agreement about the exact set of parts of speech
- From grammar school:
  - noun: *a person, place, thing, or idea*
  - verb: *a word that shows action*
  - preposition: *a word that describes relationships to a noun*
  - adjective: *a word that describes a noun*
  - adverb: *a word that describes a verb or adjective*
  - others: pronoun, conjunction, interjection

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].
  - You can't do this, for example
    - [Kim read] the book
    - [It] the book
  - See Bender #51

Context Free Grammar

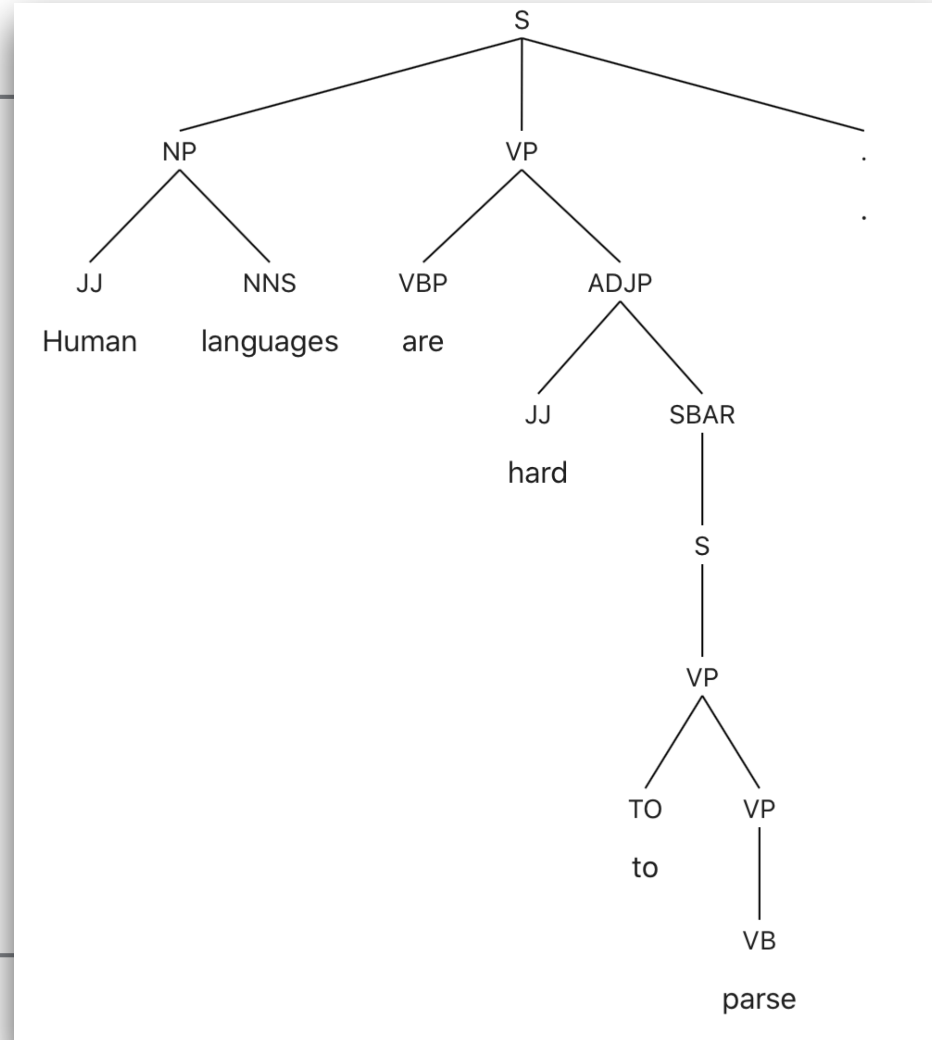
- A *finite set of* ***rules*** licensing a (possibly infinite) *number of* ***strings***
- e.g., some rules
  - [sentence] → [subject] [predicate]
  - [subject] → [noun phrase]
  - [noun phrase] → [determiner]? [adjective]* [noun]
  - [predicate] → [verb phrase] [adjunct]
- Rules are *phrasal* or *terminal*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

Chomsky formal language hierarchy refresher

regular grammar
context-free grammar
context-sensitive grammar
Turing machine

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 .



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?
  - Observe a sentence
  - *Infer* the process / structure that produced it
  - We need
    - A way to learn that model
    - An algorithm for finding the sequence of actions under that model, most likely to have produced it

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

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

× 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.

Parsing

- If the grammar has certain properties, 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

CKY algorithm

${}_0S_5 \rightarrow {}_0NP_2\ {}_2VP_5$

${}_0S_5 \rightarrow {}_0NP_1\ {}_1VP_5$

${}_1VP_4 \rightarrow VB\ PP$

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

$NP \rightarrow NN\ NN$

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

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN *NN,VB* *VB,IN* *DT* *NN*

Time *flies* *like* *an* *arrow*

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

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
- Infer the existence of a span (A, i..j) if there exists a rule A → B C and a k such that both (B, i..k) and (C, k..j) exist
- Build this chart from the bottom upward: the *opposite* direction from generation

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 iAj if iBk and kCj
```

Complexity analysis

- What is the running time of CKY
  - as a function of input sentence length?
  - as a function of the number of rules in the grammar?

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
  - (Dependency grammars—not covered today—use a simplified representation that directly model relationship between words)

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*