

- *Parsing* is the task of analyzing a string of symbols to discover its (inherent) structure
- Typically, the structure (and the valid strings in the language) is defined by a *grammar*
- The output of a parser is a structured representation of the input string, often a *tree*
- *Recognition* is an intimately related task which determines whether a given string is in a language

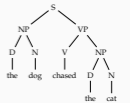
## Ingredients of a parser

(for natural language parsing)

- A formal grammar defining a language of interest
- An algorithm that (efficiently) verifies whether a given string is in the language (*recognizer*) and enumerates the grammar rules used for verification (*parser*)
- A system for ambiguity resolution (not in this course)

## Grammars

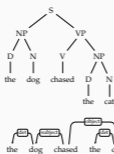
- A grammar is a finite specification of a possibly infinite language
- The most commonly studied type of grammars are *phrase structure grammars*
- Analysis using context-free grammars result in *constituency* or *phrase structure trees*



$S \rightarrow NP VP$      $NP \rightarrow D N$      $VP \rightarrow V NP$   
 $V \rightarrow$  chased     $D \rightarrow$  the     $N \rightarrow$  cat     $N \rightarrow$  dog

## Why study parsing?

- In general, it is an intermediate step for interpreting sentences
- Applications include:
  - Compiler construction
  - Grammar checking
  - Sentiment analysis
  - Information (e.g., relation) extraction
  - Argument mining
  - ...



## Different ways to represent a context-free parse



| Sentential form  | derivation             |
|------------------|------------------------|
| S                | (start)                |
| NP VP            | $S \Rightarrow NP VP$  |
| Pm VP            | $NP \Rightarrow Pm$    |
| I VP             | $Pm \Rightarrow I$     |
| I V NP           | $VP \Rightarrow V NP$  |
| I saw NP         | $V \Rightarrow$ saw    |
| I saw Pm N       | $NP \Rightarrow Pm, N$ |
| I saw her N      | $Pm \Rightarrow$ her   |
| I saw her duck N | $N \Rightarrow$ duck   |

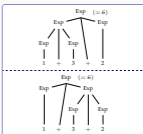
(Labelled) brackets:  $\left[ \left[ \left[ \text{I saw} \right] \text{ her} \right] \text{ duck} \right]$

## Relation between different representations

- The parse tree and the bracket representation is equivalent
  - parse trees are easier to read by humans
  - brackets are easier for computers
  - brackets are the typical representation for treebanks
- A parse tree (or bracket representation) can be obtained with a different order of production rules

## Grammars and ambiguity

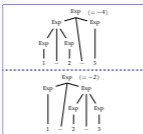
$Exp \rightarrow n$   
 $Exp \rightarrow Exp + Exp$   
(terminal symbol 'n' stands for any number)



- If a grammar is ambiguous, some sentences produce multiple analyses
- If the resulting analysis lead to the same semantics, the ambiguity is *spurious*

## Grammars and ambiguity

$Exp \rightarrow n$   
 $Exp \rightarrow Exp - Exp$

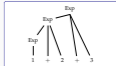


(terminal symbol 'n' stands for any number)

- Is this ambiguity spurious?
- If different structures yield different semantics, the ambiguity is *essential*

## Ambiguity can be removed from a grammar

$Exp \rightarrow n$   
 $Exp \rightarrow Exp + n$   
(terminal symbol 'n' stands for any number)

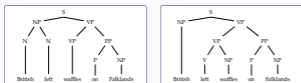


- The grammar above does not have the ambiguity of

$Exp \rightarrow n$   
 $Exp \rightarrow Exp + Exp$

- Both grammars define the same language

## Natural languages are ambiguous



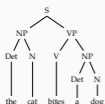
- The grammars we define have to distinguish between two different structures
- We need methods for ranking analyses

## Top-down parsing

general idea

- Start from S, find a sequence of derivations that yield the sentence
- This is simply the same as the generation procedure we discussed earlier
- Attempt to generate all strings from a grammar, but allow only the productions that 'produce' the input string

## Top-down: demonstration



|     |   |       |
|-----|---|-------|
| S   | → | NP VP |
| NP  | → | Det N |
| VP  | → | V NP  |
| VP  | → | V     |
| Det | → | a     |
| Det | → | the   |
| N   | → | cat   |
| N   | → | dog   |
| V   | → | bites |

## From demonstration to parsing

- There may be multiple productions applicable
- We need an automatic mechanism to select the correct productions
- We have two actions:
  - predict: generate a hypothesis based on the grammar
  - match: when a terminal symbol is produced, check if it matches with the one in the expected position
    - if matched, continue
    - otherwise, backtrack
- if we eliminate all non-terminals from the sentential form, and the complete input string is matched (produced), then parsing successful

## Top-down parsing: another demonstration

| the grammar | matched             | goal | production  |
|-------------|---------------------|------|-------------|
| S → NP VP   | S                   | S    | S → NP VP   |
| NP → Det N  | NP VP               | NP   | NP → Det N  |
| VP → V NP   | Det N VP            | VP   | VP → V NP   |
| VP → V      | Det N VP            | VP   | VP → V      |
| Det → a     | the                 | NP   | Det → a     |
| Det → the   | the                 | NP   | Det → the   |
| N → cat     | the cat             | N    | N → cat     |
| N → dog     | the cat             | N    | N → dog     |
| V → bites   | the cat bites       | V    | V → bites   |
|             | the cat bites       | VP   | VP → V      |
|             | the cat bites       | V    | V → bites ✓ |
|             | the cat bites       | VP   | VP → V NP   |
|             | the cat bites       | N    | N → cat ✓   |
|             | the cat bites       | VP   | VP → V NP   |
|             | the cat bites       | N    | N → dog ✓   |
|             | the cat bites a     | NP   | NP → Det N  |
|             | the cat bites a     | N    | Det → a     |
|             | the cat bites a dog | NP   | NP → Det N  |
|             | the cat bites a dog | Det  | Det → dog ✓ |

Note that the valid productions yield the parse tree.

parse: the cat bites a dog

## Top-down parsing: problems and possible solutions

- The trial-and-error procedure leads to exponential time parsing
- But lots of repeated work: dynamic programming may help avoid it
- What happens if we had a rule like NP → NP PP
- some rules may cause infinite loops
- Notice that if we knew which terminals are possible as the initial part of a non-terminal symbol, we can eliminate the unsuccessful matches earlier

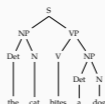
## Bottom-up parsing

## general idea

- Start from the input symbols, and try to reduce the input to start symbol
- We need to match parts of the sentential form (starting from the input) to the RHS of the grammar rules
- While top-down process relies on productions the bottom-up process relies on reductions

|            |   |       |     |       |     |     |       |           |
|------------|---|-------|-----|-------|-----|-----|-------|-----------|
| production | <table border="1"> <tr><td>NP</td><td>V</td><td>NP</td></tr> <tr><td>Det</td><td>N</td><td>Det N</td></tr> </table> | NP    | V   | NP    | Det | N   | Det N | reduction |
| NP         | V   | NP    |     |       |     |     |       |           |
| Det        | N   | Det N |     |       |     |     |       |           |
|            | <table border="1"> <tr><td>the</td><td>cat</td><td>bites</td><td>a</td><td>dog</td></tr> </table>                   | the   | cat | bites | a   | dog |       |           |
| the        | cat   | bites | a   | dog   |     |     |       |           |

## Bottom-up: demonstration



|     |   |       |
|-----|---|-------|
| S   | → | NP VP |
| NP  | → | Det N |
| VP  | → | V NP  |
| VP  | → | V     |
| Det | → | a     |
| Det | → | the   |
| N   | → | cat   |
| N   | → | dog   |
| V   | → | bites |

## A (first) introduction to shift-reduce parsing

- We keep two data structures:
  - a stack for the (partially) reduced sentential form
  - an input queue that contains only terminal symbols

|    |   |   |     |
|----|---|---|-----|
| NP | V | a | dog |
|----|---|---|-----|

- We use two operations:

shift shifts a terminal to stack

|    |   |   |     |
|----|---|---|-----|
| NP | V | a | dog |
|----|---|---|-----|

→ shift →

|    |   |   |     |
|----|---|---|-----|
| NP | V | a | dog |
|----|---|---|-----|

reduce when top symbols on stack match a RHS, replace them with the LHS of the rule

|    |   |   |     |
|----|---|---|-----|
| NP | V | a | dog |
|----|---|---|-----|

→ reduce →

|    |    |   |     |
|----|----|---|-----|
| NP | VP | a | dog |
|----|----|---|-----|

## Shift-reduce (bottom-up) parsing a demonstration

| stack     | input               | rule       | stack        | input | rule       |
|-----------|---------------------|------------|--------------|-------|------------|
|           | the cat bites a dog | shift      | NP V         | a dog | shift      |
| the       | cat bites a dog     | Det → the  | NP V a       | dog   | Det → a    |
| Det       | cat bites a dog     | shift      | NP V Det     | dog   | shift      |
| Det cat   | bites a dog         | N → cat    | NP V Det dog |       | N → dog    |
| Det N     | bites a dog         | NP → Det N | NP V Det N   |       | NP → Det N |
| NP        | bites a dog         | shift      | NP V NP      |       | VP → V NP  |
| NP bites  | a dog               | V → bites  | NP VP        |       | S → NP VP  |
| NP V      | a dog               | VP → V     | S            |       | (done)     |
| NP VP     | a dog               | S → NP VP  |              |       |            |
| NP VP     | a dog               | shift      |              |       |            |
| S         | a dog               | shift      |              |       |            |
| S a dog   |                     | Det → A    |              |       |            |
| S Det dog |                     | N → dog    |              |       |            |
| S Det N   |                     | NP → Det N |              |       |            |
| S NP      |                     | (stack)    |              |       |            |

- All input reduced to S, accept
- Rules form the parse tree

## Summary

- Parsing can be formulated as a top-down or bottom-up search (the search may also be depth-first or breadth first)
  - Naive parsing algorithms are inefficient (exponential time complexity)
  - There are some directions: dynamic programming, filtering
  - Suggested reading (for constituency parsing): Jurafsky and Martin (2009, draft 3rd ed, chapters 12 & 13)
  - A general reference for parsing: Grune and Jacobs (2007)
- Next:
- Bottom-up chart parsing: CKY algorithm
  - Suggested reading: Jurafsky and Martin (2009, draft 3rd ed, section 13.2)

