Computational Linguistics CSC 485/2501 Fall 2025

2. Introduction to syntax and parsing

Gerald Penn
Department of Computer Science, University of Toronto

Reading: Jurafsky & Martin: 5.0–1, 12.0–12.3.3, 12.3.7, [13.1–2]. Bird et al: 8.0–4.

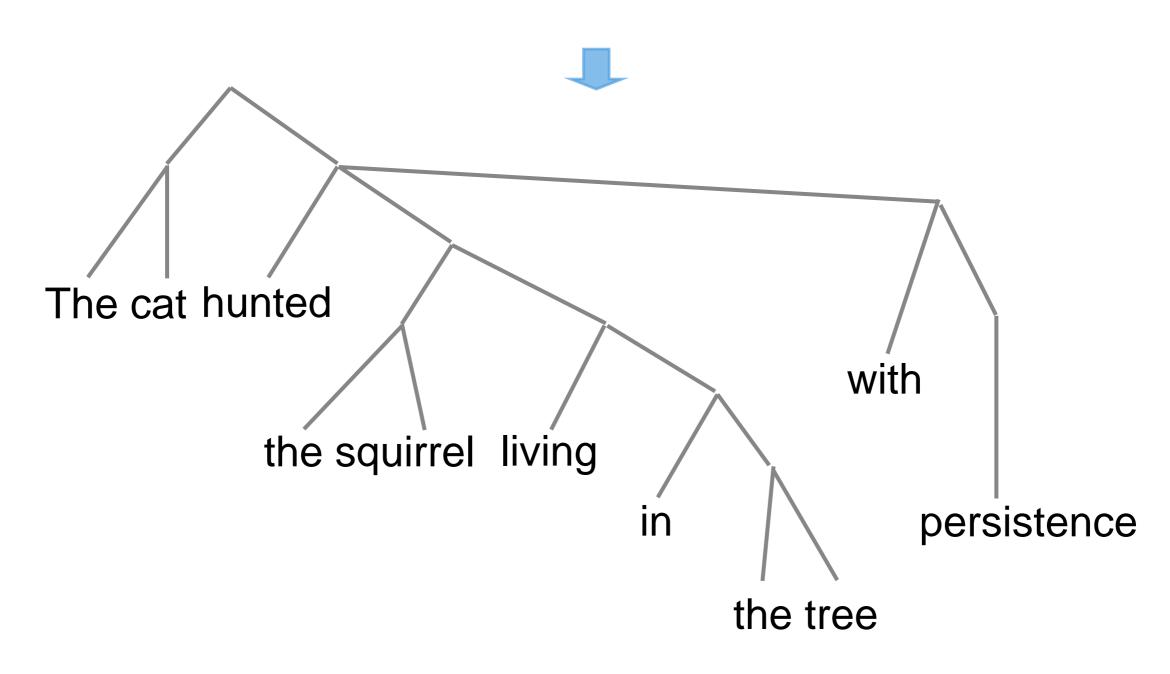
Syntax:

- The combinatorial structure of words.
- How words can be linearly organized: left/right precedence, and contiguity.
- How words can be hierarchically organized into phrases and sentences.

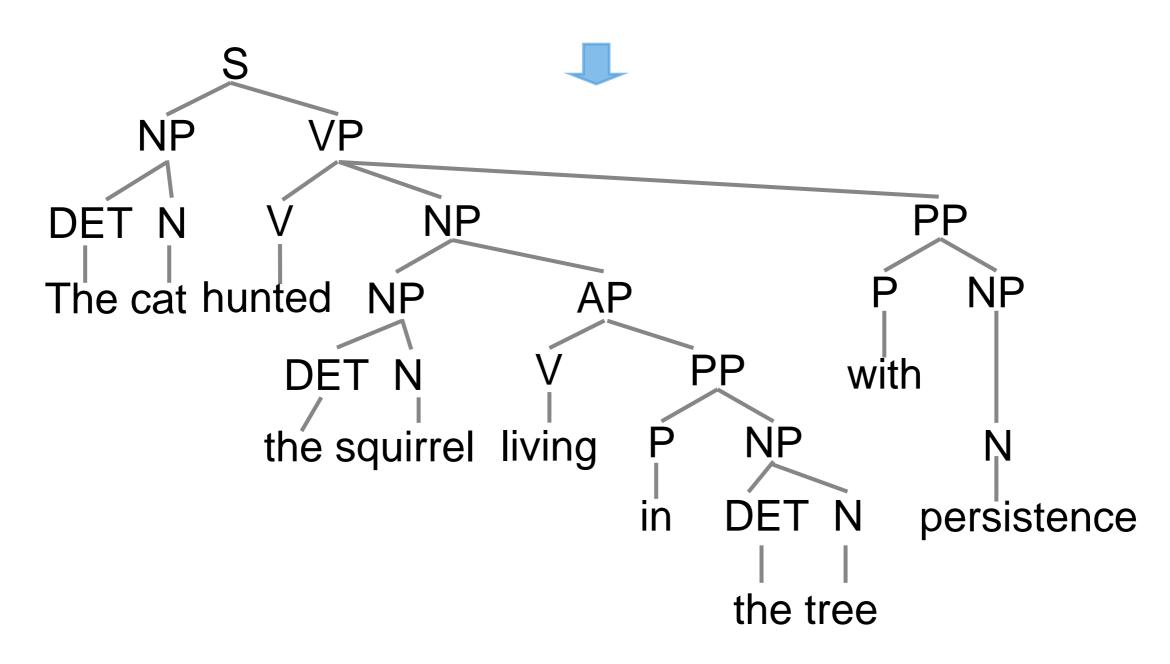
The cat hunted the squirrel living in the tree with persistence.



The cat hunted the squirrel living in the tree with persistence.



The cat hunted the squirrel living in the tree with persistence.



- Goal: meaning, interpretation, semantics.
- So why do we care about syntax?

Grammars and parsing

Grammar:

- Formal specification of allowable structures.
 - Knowledge
 - Representation

Parsing:

- Analysis of string of words to determine the structure assigned by grammar.
 - Algorithm
 - Process

Using grammar to capture structure

Main issues:

- Which words are grouped together into phrases.
- How words within a phrase project the properties of a single, common word (the head of the phrase).
- How different phrases relate to each other.
- Grammar encodes these relations. Some grammars interpret these relations with respect to meaning.

Good and bad grammars

- There are many possible grammars for any natural language.
 - Some are better than others.
- Desiderata:
 - Faithfulness to (vastly divergent) details about language.
 - Economy of description.
 - Fidelity to some prevailing linguistic intuition.
 - Efficiency of parsing.

Elements of grammar

- Primitives: lexical categories or parts of speech.
 - Each word-type is a member of one or more.
 - Each word-token is an instance of exactly one. e.g. The cat in the hat sat.
- Categories are open or closed to new words.
- Eight main categories, many subcategories.



Twenty-three

Elements of grammar

- **Primitives**: lexical categories or parts of speech.
 - Each word-type is a member of one or more.
 - Each word-token is an instance of exactly one.
- Categories are open or closed to new words.
- Eight main categories, many subcategories.



Twenty-three

 The categories might possibly be languagespecific as well.

- Nouns: denote an object, a concept, a place,
 - Count nouns: dog, spleen, Band-Aid, ...
 - Mass nouns: water, wheat, ...
 - Proper nouns: Fred, New York City, ...
- Pronouns: he, she, you, I, they, ...
- Adjectives: denote an attribute of the denotation of a noun.
 - Intersective: pink, furry, ...
 - Measure: big, ...
 - Intensional: former, alleged, ...

- Verbs: predicates, denote an action or a state. Numerous distinctions, e.g. transitivity:
 - Intransitive: sleep, die, ...
 - Transitive: eat, kiss, ...
 - Ditransitive: give, sell, ...
 - Copula: be, feel, become, ...
- Determiners, articles: specify certain attributes of the denotation of a noun that are grammatically relevant
 - the, a, some, ...

- Adverbs: denote an attribute of the denotation of a predicate.
 - Time and place: today, there, now, ...
 - Manner: happily, furtively, ...
 - Degree: much, very, ...
- Prepositions: relate two phrases with a location, direction, manner, etc.
 - up, at, with, in front of, before, ...

- Conjunctions: combine two clauses or phrases:
 - Coordinating conjunctions: and, or, but
 - Subordinating conjunctions: because, while,...
- Interjections: stand-alone emotive expressions:
 - um, wow, oh dear, balderdash, crikey, ...

Elements of grammar

- Combinations:
 - Phrase: a hierarchical grouping of words and/or phrases.
 - Clause: a phrase consisting of a verb and (almost) all of its dependents.
 - Sentence: a clause that is syntactically independent of other clauses.
- Can be represented by tree (or a labelled bracketing).
- Terminology: A constituent is a well-formed phrase with overtones of semantic and/or psychological significance.

Types of phrase 1

- Noun phrase (NP):
 - a mouse
 - mice
 - Mickey
 - the handsome marmot
 - the handsome marmot on the roof
 - the handsome marmot whom I adore
- Verb phrase (VP):
 - laughed loudly
 - quickly gave the book to Mary

Types of phrase 2

- Adjective phrase (AP):
 - green
 - proud of Kyle
 - very happy that you went
- Prepositional phrase (PP):
 - in the sink
 - without feathers
 - astride the donkey

Clauses and sentences 1

Clauses:

- Ross remarked upon Nadia's dexterity
- to become a millionaire by the age of 30
- that her mother had lent her for the banquet

Sentences:

- Ross remarked upon Nadia's dexterity.
- Nathan wants to become a millionaire by the age of 30.
- Nadia rode the donkey that her mother had lent her for the banquet.
- The handsome marmot on the roof [in dialogue].

Clauses and sentences 2

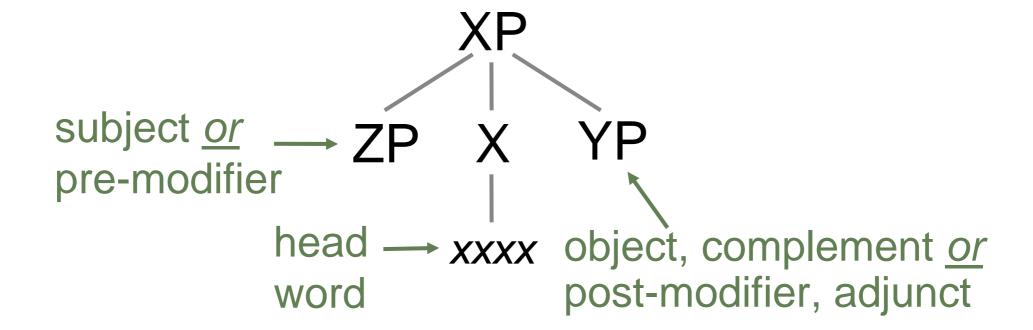
- Clauses may act as noun phrases:
 - To become a millionaire by the age of 30 is what Ross wants.
 - Nadia riding her donkey is a spectacular sight.
 - Ross discovered that Nadia had been feeding his truffles to the donkey.

Practice Quiz

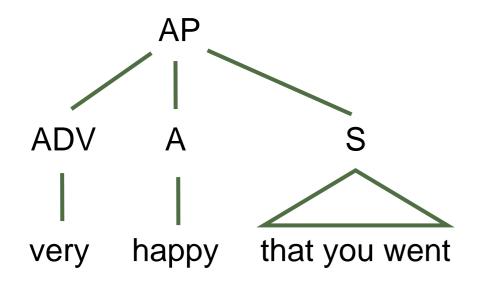
- The word that shares properties such as number with its phrase (and from which the phrase takes its name) is called the phrase's:
 - a) foot
 - b) head
 - c) centre
 - d) morpheme

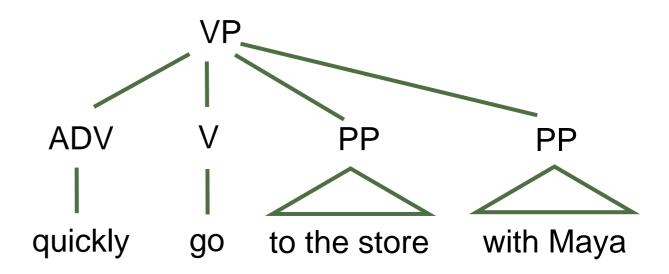
The structure of an idealized phrase

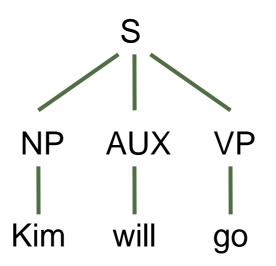
$$XP \rightarrow ZP X YP$$



Example phrases







Formal definition of a CFG

- A context-free grammar is a quadruple $G = (V_t, V_n, P, S)$, where
 - V_t is a finite set of **terminal** symbols.
 - V_n is a finite set of **non-terminal** symbols.
 - P is a finite set of production rules of the form
 A → α
 - where $A \in V_n$ and α is a sequence of symbols in $(V_n \cup V_t)^*$.
 - $S \in V_n$ is the **start** symbol.

A very simple grammar

$$S = S, P = \{ S \rightarrow NP \ VP \ NP \rightarrow Det \ N$$
 $NP \rightarrow Det \ Adj \ N$
 $NP \rightarrow NP \ PP$
 $VP \rightarrow V$

 V_t and V_n can be inferred from the production rules.

 $VP \rightarrow V NP$

 $PP \rightarrow P NP$

Det \rightarrow the | a | an

The lexicon:
In practice, a separate data structure

Lexical categories:

NT's that rewrite as a single T.

```
Adj → old | red | happy | ...
```

N → dog | park | statue | contumely | run | ...

V → saw | ate | run | disdained | ...

 $P \rightarrow in \mid to \mid on \mid under \mid with \mid ...$

Terminology

Non-terminal (NT):

A symbol that occurs on the left-hand side (LHS) of some rule.

- Pre-terminal: a kind of non-terminal located on the LHS of a lexical entry.
- Terminal (T):

A symbol that never occurs on the LHS of a rule.

Start symbol:

A specially designated NT that must be the root of any tree derived from the grammar.

In our grammars, it is usually S for sentence.

Parsing 1

- Parsing: Determining the structure of a sequence of words, given a grammar.
 - Which grammar rules should be used?
 - To which symbols (words / terminals and nodes / non-terminals) should each rule apply?

Parsing 2

- Input:
 - A context-free grammar.
 - A sequence of words
 Time flies like an arrow

```
or, more precisely, of sets of parts of speech. {noun, verb} {noun, verb} {verb, prep} {det} {noun}
```

- Process:
 - (Working from left to right?,) guess how each word fits in.

Depth-first Parsing 3

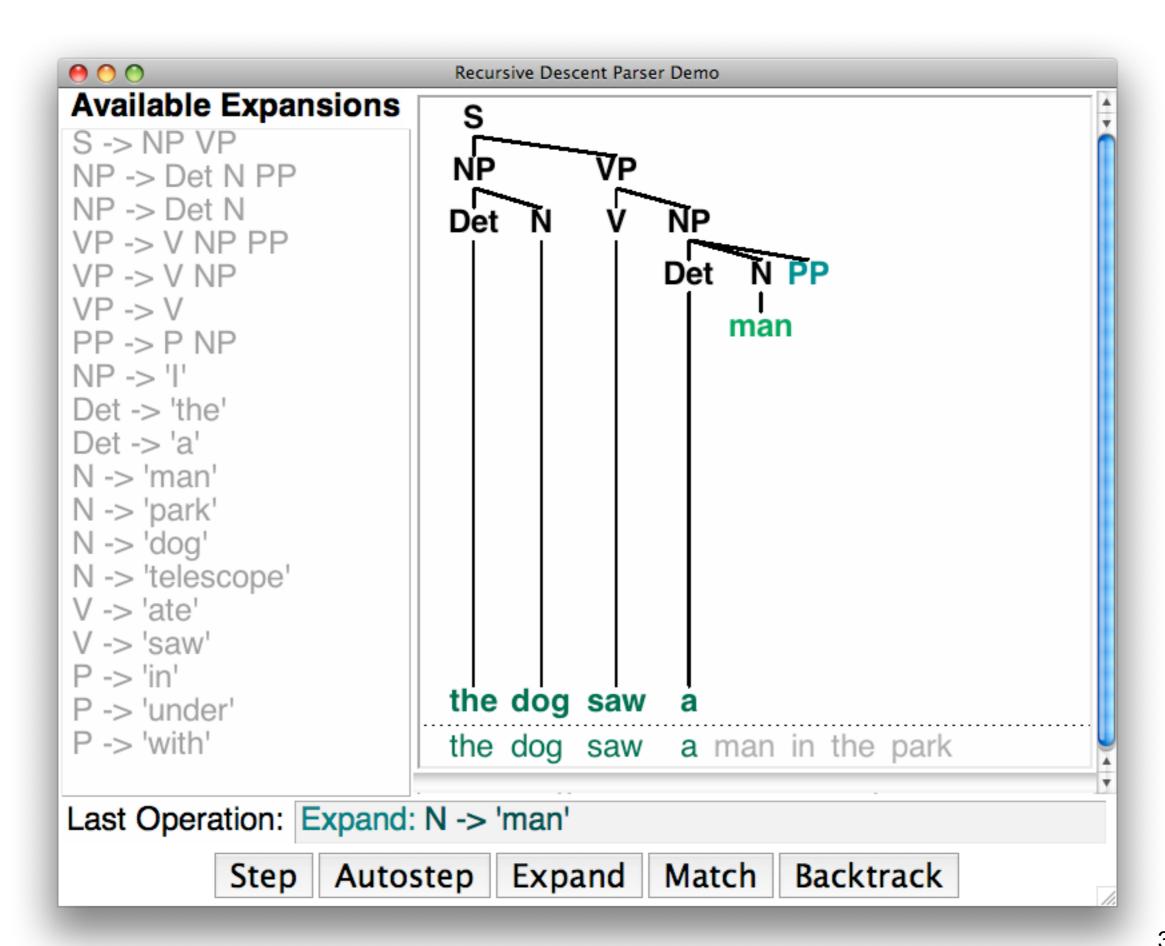
- If a guess leads to failure (parse is stymied), back up to a choice point and try a different guess.
 - Backtracking, non-determinism.
 - At each guess, must save state of parse on a stack.
 - (Or, explore in parallel.)
- Want to guess right:
 - Order of preference for rules.

Top-down parsing 1

- Top-down or rule-directed parsing: "Can I take these rules and match them to this input?"
 - Initial goal is an S.
 - Repeatedly look for rules that decompose /expand current goals and give new goals.
 E.g., goal of S may decompose to goals NP and VP.
 - Eventually get to goals that look at input.
 E.g., goal of NP may decompose to det noun.
 - Succeed iff entire input stream is accounted for as S.

Top-down parsing 2

- Example: A recursive descent parser.
 >>> nltk.app.rdparser()
- Operations on *leftmost frontier node*:
 - Expand it.
 - Match it to the next input word.



Top-down parsing 3

- Choice of next operation (in NLTK demo):
 - If it's a terminal, try matching it to input.
 - If it's a non-terminal, try expanding with first-listed untried rule for that non-terminal.

Bottom-up parsing 1

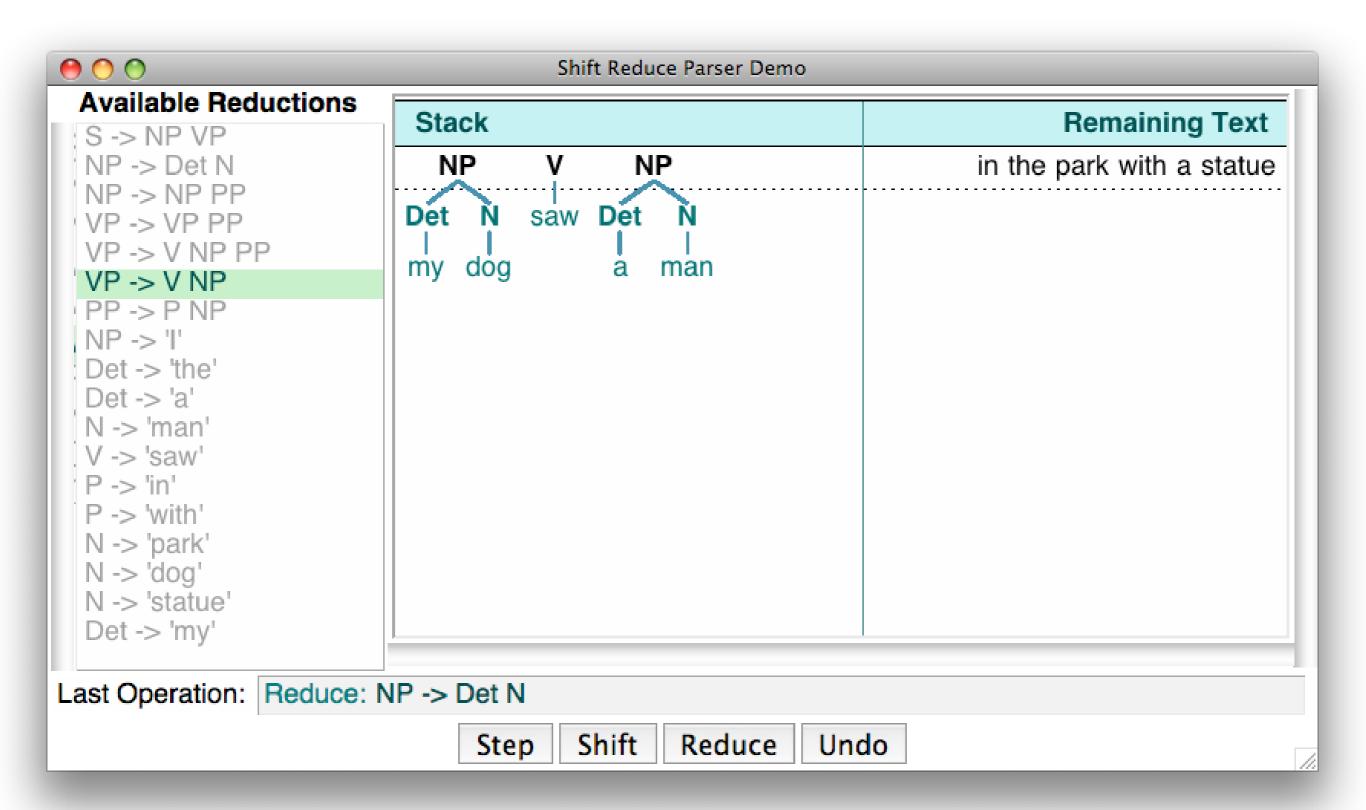
- Bottom-up or data-directed parsing: "Can I take this input and match it to these rules?"
 - Try to find rules that match a possible PoS of the input words ...
 - ... and then rules that match the constituents so formed.
 - Succeed iff the entire input is eventually matched to an S.

Bottom-up parsing 2

• Example: A shift-reduce parser.

```
>>> nltk.app.srparser()
```

- Operations:
 - Shift next input word onto stack.
 - Match the top n elements of stack to RHS of rule, reduce them to LHS.



Bottom-up parsing 3

- Choice of next operation (in NLTK demo):
 - Always prefer reduction to shifting.
 - Choose the first-listed reduction that applies.
- Choice of next operation (in real life):
 - Always prefer reduction to shifting for words, but not necessarily for larger constituents.

Problems

 Neither top-down nor bottom-up search exploits useful idiosyncrasies that CFG rules, alone or together, often have.

Problems:

- Recomputation of constituents.
- Recomputation of common prefixes.
- Solution: Keep track of:
 - Completed constituents.
 - Partial matches of rules.