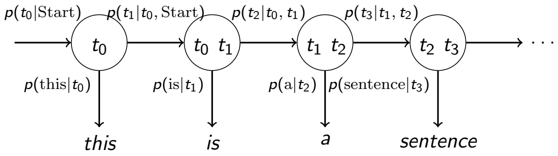
https://presemo.helsinki.fi/nlp2020

LECTURE 6: SYNTAX & PARSING

Mark Granroth-Wilding

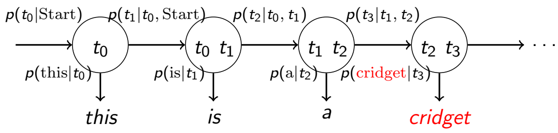
N-GRAM TAGGING MODEL

- HMM, but more states
- Markov assumption over **n-grams**
- Each **state** represents (n-1)-gram
- Parameter estimation and inference as before



DISCUSSION: DATA SPARSITY

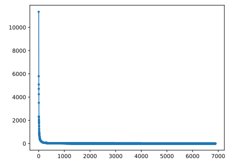
In small groups (3-4)



- **Unseen word**: how can we handle this?
- Is this sentence more of a problem?

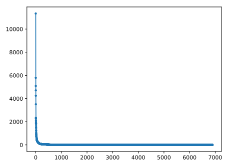
Phogen is a cridget

DATA SPARSITY



- More sparsity problems
- What to do with **unseen words**?
  - Never seen: **default tags**
  - Probabilities of tags across corpus
  - Only affects **emission distribution**
  - → use a tag that looks right in context
  - Problems if several unknown words in sentence
  - Is there **some** information we can use?
    → Morphology?

DATA SPARSITY



- More sparsity problems
- What to do with **unseen tag contexts**?
  - Can't estimate $p(t_i|t_{i-1}, t_{i-2})$ for unseen $t_{i-2}t_{i-1}$
  - Unreliable if only seen few times
  - Problem for *higher-order* models ($n > 2$)
  - Solutions: **smoothing/backoff**, as with LM

MORE DATA: UNSUPERVISED LEARNING

- Can train HMM with **unsupervised learning**
- Advantage: use **lots of data**, any **language**, **domain**, without annotation
- Disadvantage: no real POS tags, just *word clusters*
- States might not correspond to our POS tags
- Can still be useful
- Train HMM with **expectation-maximization** (EM)

MORE DATA: SEMI-SUPERVISED LEARNING

- **Semi-supervised**:
  - some annotated data
  - expand model with more raw data
- POS tags based on annotations
- Initial distributions estimated on annotations
  - tag raw data using basic model
  - re-estimate model
  - iterate

PROBABILISTIC MODELS

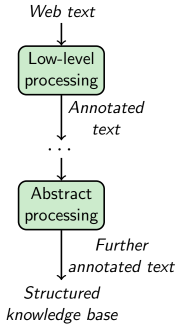
More probabilistic models coming up today

- Similar model
- **Structured** prediction for syntax

MACHINE LEARNING IN THE PIPELINE

- Statistical POS tagging: applied **ML** to single pipeline component
- Statistical models in other components
- ML applied to many NLP sub-tasks
- Almost no purely rule-based systems today
- Hand-crafted rules & statistical models in different combinations

E.g. Hand-crafted syntactic grammars, with statistical parsing models



OTHER STATISTICAL METHODS

- These statistical methods fairly simple
- More advanced methods exist in all areas
- Big strides in ML (outside NLP)
- Exploit lots of unannotated data
- Apply **in pipeline**
  - Components exploit more data
  - Unannotated data?
  - Advanced models/learning algorithms
  - E.g. parsing using NNs
- Might not use traditional pipeline
- E.g. *character-level RNNs* for sentiment analysis: skip tokenization, POS tagging, parsing, . . .

(big)

ADVANCED STATISTICAL METHODS

Later in course:

- Bayesian modelling
- Deep learning
  - Recurrent neural networks (RNNs)
  - Convolutional neural networks (CNNs)

More in lecture 13

FORGETTING THE PIPELINE

- Example: **neural machine translation** (NMT)
- Drops whole pipeline
- Learns input → output mapping
- Doable for some other tasks
- But: when it goes wrong, it can go very wrong
- Hard to analyse & fix errors
- *More in NLG lectures*

STRUCTURE IN LANGUAGE

- NL sentences have structure
- Hidden: expressed as a sequence

Agreement in number

Alice walk**s** quickly
Athletes **walk** quickly

- Dependencies between words
- May not be adjacent
- Can be arbitrarily far apart

STRUCTURE IN LANGUAGE

- NL sentences have structure
- Hidden: expressed as a sequence

Alice, with her big boots on, walk**s** quickly
Athletes, after much training, **walk** quickly

- Dependencies between words
- May not be adjacent
- Can be arbitrarily far apart

STRUCTURE IN LANGUAGE

- NL sentences have structure
- Hidden: expressed as a sequence

Alice, with her big boots on, walk**s** quickly
Alice, with her big boots on, seeing the man without a hat on on top of the hill, walk**s** quickly

- Dependencies between words
- May not be adjacent
- Can be arbitrarily far apart

LONG-RANGE DEPENDENCIES

Alice walk**s** quickly
Alice, with her big boots on, seeing the man without a hat on on top of the hill, walk**s** quickly

- Need models that capture these to understand language
- **Long-range** dependencies (and others)
- Much language processing depends on this
- Linguistic sequence ⇒ meaning: infer hidden connections & relationships
- **Syntax** models *structures* underlying this *process*

EXERCISE: DEPENDENCIES

In small groups

Alice, who saw the man, who pushed Bob,
who ate the apple, walks quickly.

- Write out this sentence
- Draw lines indicating **dependencies**,
  or connections in the underlying sentence structure
- Group words / elements visually as you see fit

SYNTAX: SUBSTITUTABILTY

- POS tags capture type of **substitutabilty** of words:

Alice walks quickly
John walks quickly
Alice walks occasionally

- **Syntactic categories** capture substitutabilty at **phrasal level**

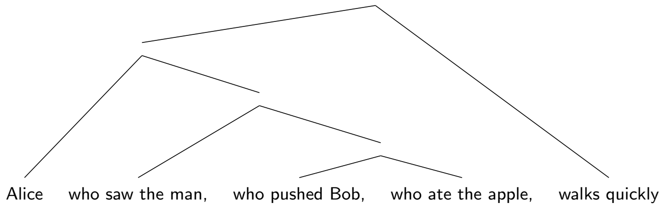
Alice, with her boots on, walks quickly
Alice, without a second thought, walks quickly
Alice, who longs to get home, walks quickly

SYNTAX: RECURSION

- (Probably) all human languages exhibit **recursion**

Alice, who saw the man, who pushed Bob,
who ate the apple, walks quickly

- Natural to analyse as a **tree structure**:



FORMAL SYNTAX

- Theory of syntax: characterizes **well-formed** sentences

	Alice walks quickly
*	Alice walks red

- Early formal syntax focussed on this
- **Well-formed**: conforms to rule of language's grammar
- Rules: produce **semantic interpretation**
- But can follow *syntax* rules with being meaningful:

Colorless green ideas sleep furiously

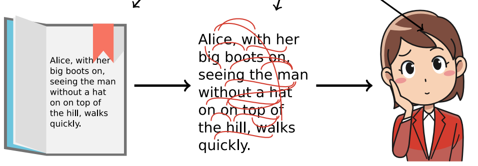
Noam Chomsky, *Syntactic Structures* (1957)

FORMAL SYNTAX

Rules: produce **semantic interpretation** by **composition** of parts

Syntax describes *process* mapping
surface form to *meaning*
via analysis of *hidden structure*

Parsing



GRAMMAR FORMALISMS

- *Formal language* to capture hidden structure of NL
- There are **many** formalisms
- Two widely used ones:
  1. Context-free grammar (CFG) ← Now
  2. Dependency grammar ← Next lecture
- Need:
  - Grammar for language
  - Algorithm to **parse**
  - Statistical model / data

Analyse **structure**
using **grammar**
for given surface form

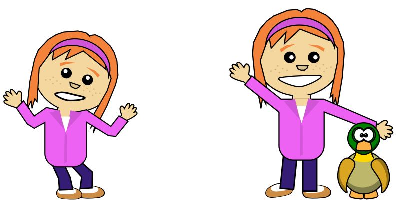
AMBIGUITY

parsing

- Syntactic structure is ambiguous
- One sentence can have many structures
- Inferring structure can disambiguate meaning

POS ambiguity

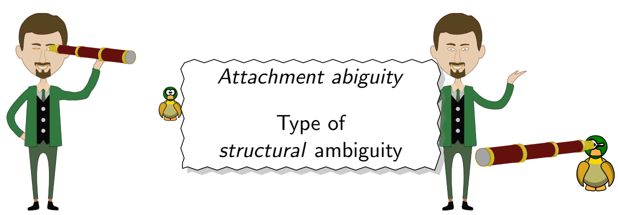
I saw her duck



DISAMBIGUATION: ATTACHMENT

- Prepositional phrase: *with the telescope*
- What does it **attach** to?

I saw the duck with the telescope



CONTEXT-FREE GRAMMARS: REMINDER

Grammar consists of:

1. Set of **terminal** symbols ← Words / vocabulary *duck, man*
2. Set of **non-terminal** symbols ← Phrase categories S, NP, etc
3. Set of **rules** (productions)
   single NT → sequence of NTs / terminals
4. Start symbol ← Usually S *(sentence)*

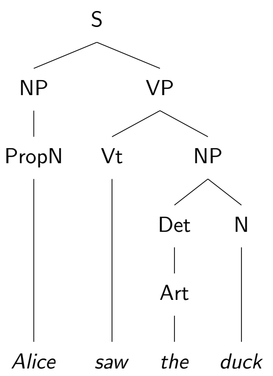
Generative grammar:
words of sentence *generated* from *start symbol* via *productions*

A SMALL CFG

Rules
S → NP VP
NP → Det N
NP → PropN
Det → PosPro
Det → Art
VP → Vt NP

Lexicon
Art → the | a
PropN → Alice
N → duck | telescope | park
Vt → saw
PosPro → my | her

EXAMPLE SENTENCE DERIVATION

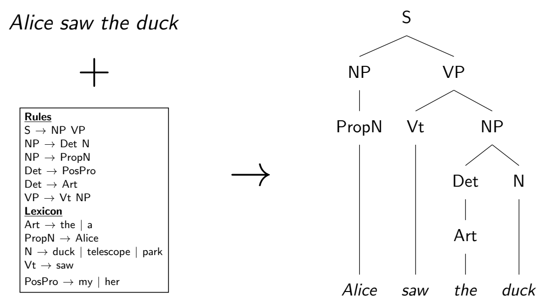


Rules
S → NP VP
NP → Det N
NP → PropN
Det → PosPro
Det → Art
VP → Vt NP

Lexicon
Art → the | a
PropN → Alice
N → duck | telescope | park
Vt → saw
PosPro → my | her

PARSING

- **Generated** words of sentence from S
- Other way: **analyse** possible underlying structures – **parse**



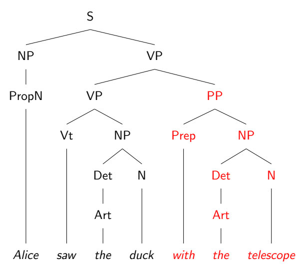
SYNTACTIC AMBIGUITY

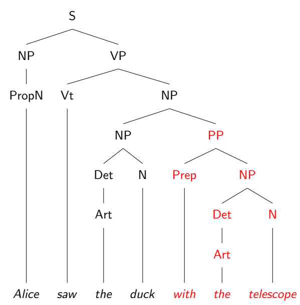
- Can be many underlying structures for one sentence
- NL parsers are **non-deterministic**
- Unlike programming languages:
  - Unambiguous structure
  - Deterministic parser

I saw the duck with the telescope

- *Some* 'real' ambiguities – perceived by humans
- Others: *overgeneration* by the grammar

AMBIGUOUS STRUCTURES





EXERCISE: PARSING

In small groups

Rules
S → NP VP
NP → Det N
NP → PropN
Det → PosPro
Det → Art
VP → Vt NP
VP → VP PP

Lexicon
Art → the | a
PropN → Alice
N → duck | telescope | park
Vt → saw
PosPro → my | her
Prep → in

Draw a derivation tree, under this grammar, for:

The duck saw my telescope in the park

PARSING

String + Grammar → Syntactic structures (trees)

- Potentially many possible trees
- *All possible derivations* using grammar for given string
- Capture structures important to **semantics**
- Coming up:
  algorithm to find *all* derivations: **exhaustive parsing**

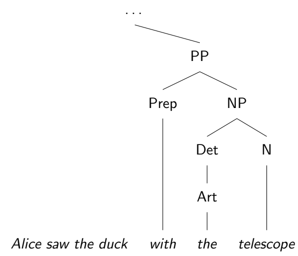
BOTTOM-UP PARSING

- Grammar **top-down**: S ⇒ words
- Parsing **bottom-up**: words ⇒ tree (S)
- Bottom-up strategy:
  - Consider all *NTs* that could have generated a *word*
  - Consider all NTs that could have generated *that*
  - . . . until S covering whole sentence
- May be multiple full **derivations**

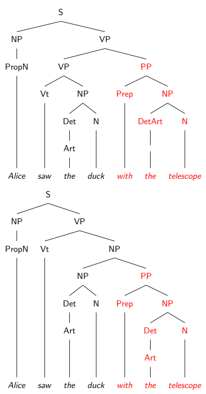
BOTTOM-UP PARSING

Dynamic programming approach:

- Rule is **context free**
- Doesn't depend on **subtree** or **context outside**
- Same NT for same span: treated same further up
- Remember all derivations to retrieve at end



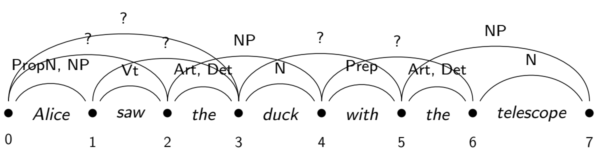
CFG WITH PREPOSITIONAL PHRASES



Rules
S → NP VP
NP → Det N | PropN
Det → PosPro | Art
VP → Vt NP
VP → VP PP
NP → NP PP
PP → Prep NP

Lexicon
Art → the | a
PropN → Alice
N → duck | telescope | park
Vt → saw
PosPro → my | her
Prep → with

CHART PARSING



THE CKY ALGORITHM

(AKA: CYK algorithm)

```
for len = 1 to n:                # Num words in span
    for i = 0 to n-len:          # Start of span
        j = i+len                # End of span
        # Apply any unary rules
        foreach A->B where B in c[i,j]:
            Add A to c[i,j]
            for k = i+1 to j-1:  # Middle position
            # Process binary rules covering sub-spans
            foreach A->B C where B in c[i,k] and C in c[k,j]:
                Add A to c[i,j]
```

- Goal: S covering the whole sentence (`c[0,n+1]`)
- Store traces to retrieve all trees

CKY

- This version strictly bottom-up (depth-first)
- Can change ordering to make more **incremental** ← Left-to-right
  See version in *J&M*
- Grammar must be in **Chomsky normal form**
  - Can convert any CFG to CNF

EFFICIENCY OF CKY

- Dynamic programming:
  avoid *recomputing* unnecessary structures
- Still computes many unused structures
- Other algorithms exist, motivated by
  - Cognitive modelling
  - Average time efficiency
  - Space efficiency, . . .
- Another approach: **statistical parsing**
- Next lecture. . .

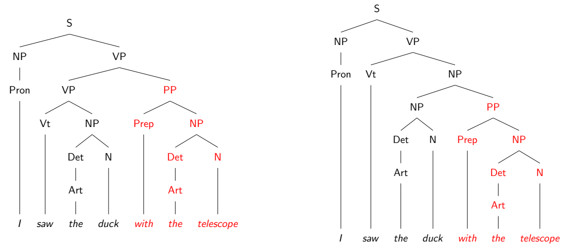
AMBIGUITY

- Parsing ambiguity a big problem
- **Correctness**: finding correct structure from *many* possible
- **Efficiency**: many structures to consider → parsing is slow

I saw the duck with the telescope

- Real ambiguities (perceived by humans) – want these
- *Overgeneration* by grammar – as few as possible

PARSING FOR SEMANTICS



- *Some* **parsing ambiguities** $\Rightarrow$ ambiguities of **meaning**
- Parsing helps resolve ambiguity
- Choosing single parse tree, some may remain

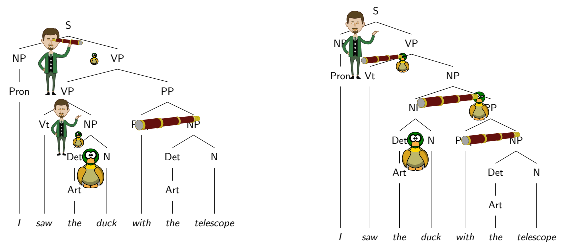
PARSING FOR SEMANTICS

- Representing **meaning** of sentences: **semantics**
- Putting together meanings of *parts of sentence* to produce whole meaning:

  compositional semantics

  - Sometimes meaning is **non-compositional**
  - *Kick the bucket* $\neq$ *kick* + *bucket*
- Syntax: how to combine parts
- Ambiguities lead to different combinations

PARSING FOR SEMANTICS



- Represent meaning using **formal logic** (as we've seen)
- Bits of sentence: bits of logic
  - Often with unfilled 'holes'
- Follow parse tree to combine bits

PARSING FOR SEMANTICS

- Process of **composition** main reason for parsing
- No time for details now
- Not all semantic representations are *logical*
- May still need to follow structure of composition

FSTs FOR SYNTAX?

Question: why cant we use FSTs?

this sentence (is (not) well) bracketed) .

?

Alice, with her big boots on, seeing the man without a hat on on top of the hill, walks quickly

FSTs FOR SYNTAX?

Consider language:

$$\{a^n b^n\}$$

ab, aabb, aaabbb, ...

Not: aaab

- Is this language **regular**?
- Can we write a **regex** or **FSA** for it?
- Why?

EMBEDDING

But we can write a CFG:

X → aXb
X → ab

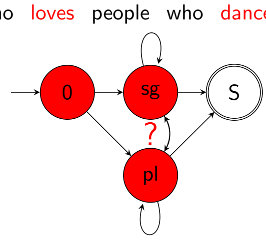
abs must be **embedded** in phrases
Closely related to **embedding** in NL syntax:

Alice, with her big boots on [etc], walks quickly

Processor must **remember** how many *a*s appeared before *b*s

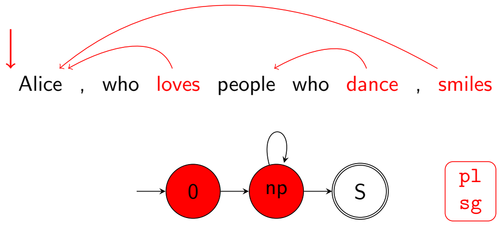
EXTENDING AN FSA

Alice , who loves people who dance , smiles



- FSA captures: **subject-verb number agreement**
- Need to remember *Alice*=sg on reaching *smiles*
- But another (nested) pair intervened

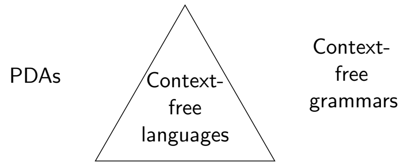
EXTENDING AN FSA



- Add a **stack** memory: *push*, *pop*, *peek* at top
- Maintain `sg` feature until `pl` pops off

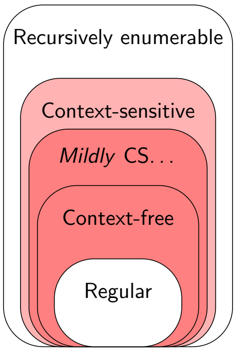
PUSH-DOWN AUTOMATON

FSA + stack = *push-down automaton* (PDA)



THE CHOMSKY HIERARCHY: SYNTAX

- Expressivity to capture recursion, hierarchical structure
- *At least* CF power
- Enough for *most* language use
- A *few* phenomena in many languages exceed CF
- Other formalisms capture this *CCG, TAG, HG, LIG*, ...



STRONG VS WEAK EQUIVALENCE

Two grammars *equivalent* if they generate same languages

- **Weakly equivalent**: same set of strings
- **Strongly equivalent**: same strings *and* same underlying structures: bracketing/dependencies

SUMMARY

- POS tagging
  - Data sparsity
  - Semi-/unsupervised learning
- Statistical models
  - No (traditional) pipeline?
- Sentence structure: syntax
  - Formal syntax
  - CFGs
  - Bottom-up parsing: CKY
  - Semantic parsing: sentence → semantics

Next lecture: **statistical parsing**

READING MATERIAL

- HMMs for POS tagging: *J&M3 8.4*
- More detail on HMMs: *J&M3 appendix A*
- Syntax, CFGs, treebanks: *J&M3 10.1-4, Eisenstein ch 9*
- Parsing, CKY: *J&M3 ch 11, Eisenstein 10.1-2*