# Linguistic and Computational Theories of Grammar

## LCTG—Notes 1: The problem of Natural Grammar

*Given the grammar of a language, one can study the use of the language statistically in various ways; and the development of probabilistic models for the use of language (as distinct from the syntactic structure of the language) can be quite rewarding.*

Chomsky, 1957:17, note 4



## The Problem of NLP

- In 1949, Shannon and Weaver published *The Mathematical Theory of Communication*, showing that statistical approximations to English based on Markov processes could be used to encode English efficiently for transmission in noise. Tasks like machine translation could be solved by treating e.g. Russian as a noisy encoding of English.
- In 1957, Chomsky published *Syntactic Structures*, showing that natural grammars could not be exactly captured by such methods. It seemed to follow that machine translation could not be modelled as a noisy channel (although the machines were too small to actually try any of this).
- Chomsky made a point of being open to the idea that statistics could guide grammatical *processing.* (See quote above).
- Nevertheless, most work in computational linguistics switched to linguistically informed high-level representations of syntax and semantics, and small knowledge domains (e.g. the LUNAR project).



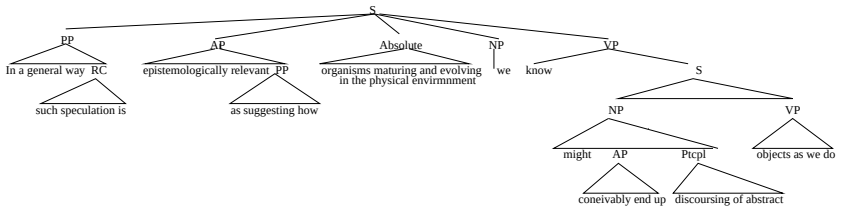
## The problem (Contd.)

- Around 1988, the machines got big enough to try both techniques.
- Surprisingly, low level approximations such as Markov processes worked better than handbuilt linguistically informed representations on almost all tasks, such as speech recognition, parsing free text, and, eventually, MT itself.
- The reason was twofold:
  - The Search Space for realistically sized grammars is huge due to syntactic ambiguity
  - Everything in Natural Language obeys a Power-Law distribution a.k.a. Zipf's Law



## The Problem: Ambiguity

- "In a general way such speculation is epistemologically relevant, as suggesting how organisms maturing and evolving in the physical environment we know might conceivably end up discoursing of abstract objects as we do." (Quine 1960:123, cf. Abney 1996).
- —yields the following, among many other horrors:



# The Problem: Zipf's Law

- A Zipfian distribution is one in which the frequency of any event is about twice that of the next most frequent event.
- For example, the word "the" is the most frequent word in written text at around 7% of the total. "Of" is the next most frequent word, at around 3.5%
- Such doubly exponential functions plot as a (mainly) straight line on a log-log graph.
- Zipf's Law means that you can get 80% of the variance by modeling the most frequent events, and ignoring the "long tail" of exponentially rare events entirely.
- However, Zipf's Law *also* means that *all the information about the system* is in the "long tail" of exponentially rare events.
- **Machine Learning is very bad at learning systems where all the information is in rare events.**



# The Problem (Contd.)

- Most work in computational linguistics switched to linguistically uninformed, low-level statistical approximations and machine learning.
- Around 2000, the process began of putting linguistics, statistical models, and machine learning, back together again.
- When Moore's Law gives out, and the machines stop getting exponentially bigger, as they must, knowing linguistics will be essential for computational linguistics to advance.
- This course will teach you about the properties of language wyou need to know about in order to face that problem



# The Readings: Shannon 1948; Chomsky 1957

- Chomsky's 1957 book *Syntactic Structures*, together with certain more technical papers from around the same time, is one of the most important documents in linguistics and cognitive science.
- The theory in detail has been completely superseded.
- Nevertheless, surprisingly many of the formal devices that it includes recur, particularly in the most modern descendents, including: kernel sentences (aka lexicalized elementary trees or categories etc., notes 7); generalized or "double-based" transformations (aka Merge, combinatory rules, tree-adjunction, etc., notes 6-7); affix-hopping (notes 11); the role of statistics in natural language processing (notes 9).
- Rather than presenting this first theory of grammar in detail, we will look at the entire development that it led to.



# The Readings (Contd.)

- One crucial ingredient of the theory was a hierarchy of language types, each type characterized by a class of rules that are sufficient to specify all languages of that type, an automaton which is sufficient to recognise whether a sentence is from a given language of that type, and a class of languages including all those of classes lower in the hierarchy as a proper subset.
- The Shannon paper represents an earlier, information-theoretic view of language, which to some extent Chomsky was reacting against. It introduces the notion of a statistical model as a measure of how probable a string or an analysis is, given some corpus of the language.
- *As Chomsky pointed out* (1957:17), such notions become very important in natural language processing by machine with grammars of the size and degree of ambiguity that human beings actually work with, because of the huge search problems that such parsers encounter (notes 9).

# The Readings (Contd.)

- The first three chapters of Chomsky 1957 show that human languages fall outside the lowest level of **Finite State** languages, and are *at least* at the level of **Context Free** Languages.
- The proof requires a distinction between ideal linguistic capacity, now known as **Competence** and the **Performance** mechanism that actually processes sentences.
- Competence allows sentences that are so long or convoluted that none of us will live long enough or have enough memory to process them. Performance cannot cope with them—but clearly this limitation is accidental, not a fact about English.
- *Syntactic Structures* goes on to suggest that the level of human grammars is still higher in the hierarchy. It raises (but does not answer) the question of which level is just high enough to contain all human languages.



# The Readings (Contd.)

- One of the purposes of the course is to attempt an answer to this question, following work by Joshi and colleagues.
- The course will proceed by putting together components from a number of theories that followed *Syntactic Structures*, showing what problems motivated each innovation, and attempting to formulate a unified synthesis of all of them, with an emphasis on practical implemental systems.
- One crucial ingredient is already implicit in the Shannon paper. When we come to the problem of parsing with coverage comparable to human readers of texts like books and newspapers—that is, to the question of performance, in Chomsky's terms—we will find that probabilistic models of the kind considered in Shannon 1948 are crucial.



# The Chomsky Hierarchy

| Language Type | Automaton | Rule-types | | | Exemplar |
|---|---|---|---|---|---|
| Type 0: RE | Universal Turing Machine | $\alpha$ | $\rightarrow$ | $\beta$ | |
| Type 1: CS | Linear Bound Automaton | $\phi A \psi$ | $\rightarrow$ | $\phi \alpha \psi$ | $a^{2^n}$ |
| Type ?: I | Nested Stack Automaton | $A_{[(i),\ldots]}$ | $\rightarrow$ | $\phi B_{[(i),\ldots]} \psi$ | $a^n b^n c^n$ |
| Type 2: CF | Push-Down Automaton | $A$ | $\rightarrow$ | $\alpha$ | $a^n b^n$ |
| Type 3: FS | Finite-state Automaton | $A$ | $\rightarrow$ | $\begin{cases} \text{a B} \\ \text{a} \end{cases}$ | $a^n$ |



# The Chomsky Hierarchy (Contd.)

- $a, b, etc.$ are terminal symbols i.e. the atomic symbols of the language (words, in English);
- $A, B, etc.$ are nonterminals, i.e. the phrasal types of the language (e.g. NP and VP);
- $\alpha, \beta, etc.$ are any string of terminals and nonterminals, including the empty string.

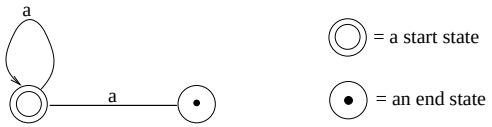
# Grammars Languages and Trees

- A grammar is defined as a set $T$ of terminals; a set $N$ of non-terminals; a set $R$ of rules; and a start symbol $S$ drawn from $T$.
- A grammar so defined is said to "generate" a language defined as the (usually infinite) set of strings of terminals that you get by starting with the start symbol, and applying all possible rules, and then applying all possible rules to the nonterminals in the resulting strings, and so on.
- For any given string in the language this procedure assigns one or more "derivations" or sequences of rule applications, usually represented as trees. Trees are strongly related to the process of semantic interpretation.
- The languages, automata, and grammars characteristic of any given level of the hierarchy subsume those at lower levels, in the sense that the set of languages at that level *properly includes* the sets of languages at lower levels, the automaton can simulate the lower level automata, and the set of possible rule-types properly includes all the lower level rule-types.

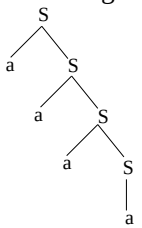


# Type 3: Contd: FSA

- The following *finite-state automaton* (FSA) generates the finite-state/regular language $a^n$ consisting of strings of any number $n$ occurrences of the word or symbol $a$, where $n \geq 1$:



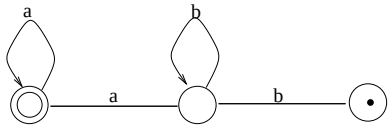

- It corresponds to the regular grammar consisting of the two rules $S \rightarrow a\ S$ and $S \rightarrow a$, and gives rise to the following tree for the string *aaaa*:






# Type 3: Finite-State Automata/Regular Grammars

- The following FSA generates the language $a^n b^m$—$n$ $a$s followed by $m$ $b$s, $m, n \geq 1$:




(What is the corresponding regular grammar? What tree does it yield for *aabbb*?)

- If we attach probabilities to the arcs of a FSA then we get a device called a Probabilistic Finite State Machine (PFSA) or Markov process. A Markov process is first order if the probabilities depend only on the current state of the automaton, a property called *ergodicity*.
- A Markov process whose probabilities depend only on a sequence of $n$ previous states including the present state is an $n$th-order Markov process. (The definition of ergodicity can be correspondingly generalized.)

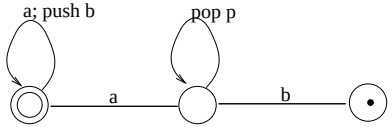


# Type 3: Contd: PFSA/Markov Process

- The Shannon 1948 paper shows that, when state-transitions are associated with the emission of letters or words, surprisingly good approximations to English can be obtained with quite low-order Markov processes, a fact that becomes important when we consider practical natural language processing by machine.
- Third-order Markov approximation to English letter transition probabilities:
  - IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.
- Second-order Markov approximation to English word transition probabilities:
  - THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

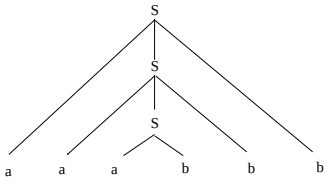
# Type 2: PDA/Context-Free Grammars

- However, there are properties of language that cannot be fully captured by FSAs or Markov processes of any order.
- No FSA can generate the language $a^n b^n$—strings containing one or more *a*s followed by *the same number of bs*.
- (Adding a counter to the previous FSA makes it no longer an FSA.)
- We need a new kind of machine called a *Push-Down Automaton* (PDA). A PDA is a FSA plus a last-in first out memory called a "stack". Transitions can not only emit or consume a symbol, but can also "push" items onto the top of the stack or "pop" them off the top of the stack. The following automaton generates $a^n b^n$, $n \geq 1$, given a general condition that the stack must be empty on termination:






It corresponds to the context-free grammar $S \rightarrow a\ S\ b$ and $S \rightarrow ab$, and gives rise to the following tree for the string *aaabbb*:




Note that the **dependencies** between pairs of *a*s and *b*s stemming from each successive application of the rule are **nested**.

- If we attach probabilities to the productions of a context-free grammar, then we get a probabilistic context-free grammar (PCFG).
- PCFGs thus interpreted give a surprisingly bad approximation to English and other languages, because the independence assumptions that they embody (such as that the way the daughter NP of a VP gets expanded is independent of the way the V gets expanded) are wrong.
- We will see a way round this when we look at probabilistic parsing.



# Strong and Weak Equivalence

- Two grammars are said to be "weakly equivalent" if they generate the same language or stringset.
- Two grammars are "strongly equivalent" if they assign the same tree(s) to each string in the same language.
- All grammars at a given level in the hierarchy have strongly equivalent grammars at higher levels, but not vice versa.
- A grammar or class of grammars is said to be "strongly adequate" to the capture of a language or class of languages if it assigns the "right" trees to strings. The "right" tree is the one we need for semantic interpretation.
- Weakly equivalent grammars which assign the "wrong" trees are said to be only "weakly adequate."



# Expressive Power of Human Grammars

- Chomsky proved on the basis of the recursive or embedding properties of human languages that strongly adequate human grammars required at least context-free power.
- The following family of sentences in English has the property of embedding propositions within propositions to an unbounded depth, as indicated by the brackets:
  a. I saw [Harry swim].
  b. I saw [John see [Harry swim]].
  c. I saw [Anna help [John see [Harry swim]]].
  d. etc. etc.
- However, right periphery embedding is a regular grammar, hence weakly finite-state.

# On Peripheral Recursion

- Notice that for the semantics, we want the recursion, i.e. as linguists we want to think of this as context-free.
- Being weakly finite state simply means that there is a way to compute the semantics that is iterative as in programming language tail-recusion.



# *At Least* Context-Free Power

- The crucial case is in fact *center* embedding, for which there is not even a weakly adequate finite-state grammar. So a better example is the corresponding German family:

  a. daß ich [Heinrich schwimmen] sah.

  that I Heinrich swim saw
  "that I saw Henry swim"

  b. daß ich [Johannes [Heinrich schwimmen] sehen] sah.

  that I Johannes Heinrich swim see saw
  "that I saw John see Henry swim"

  c. daß ich [Anna [Johannes [Heinrich schwimmen] sehen] hilfen] sah.

  that I Anna Johannes Heinrich swim see help saw
  "that I saw Anna help John see Henry swim"

  d. etc. etc.

- Is Context-Free Grammar adequate for all human language?



# *More than* Context-Free Power

- Chomsky argued convincingly that CFG was not strongly adequate.
- For example, a strongly adequate grammar for Dutch seems to require more than context-free power:

  a. dat ik$_1$ Henk$_2$ zag$_1$ zwemmen$_2$.

  that I Henk saw swim
  "that I saw Henry swim"

  b. dat ik$_1$ Jan$_2$ Henk$_3$ zag$_1$ zien$_2$ zwemmen$_3$.

  that I Jan Henk saw see swim
  "that I saw John see Henry swim"

  c. dat ik$_1$ Anna$_2$ Jan$_3$ Henk$_4$ zag$_1$ helpen$_2$ zien$_3$ zwemmen$_4$

  that I Anna Jan Henk saw help see swim
  "that I saw Anna help John see Henry swim"

  d. etc. etc.

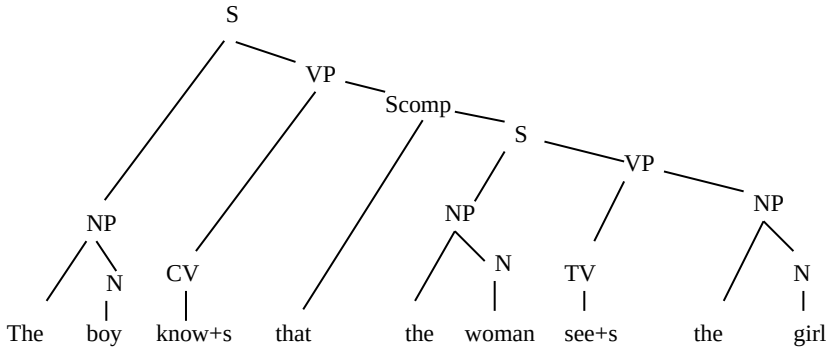
- What power *do* we need? The less we can get away with the better.



# A Toy CFPS Grammar for a Fragment of English

$$
\begin{array}{rcl}
S & \rightarrow & NP \quad VP \\
NP & \rightarrow & \begin{cases} \text{the} \quad \text{N} \\ \\ \{John, Fred, Sally, Mary, \ldots\} \end{cases} \\
VP & \rightarrow & \begin{cases} \text{TV} \quad \text{NP} \\ \text{CV} \quad \text{Scomp} \end{cases} \\
Scomp & \rightarrow & that \quad S \\
N & \rightarrow & \{man, woman, boy, girl, \ldots\} \\
TV & \rightarrow & \{see, like, loath, \ldots\} + s \\
CV & \rightarrow & \{see, know, dream, \ldots\} + s
\end{array}
$$

# A Derivation




- The structure of the tree reflects the structure of the interpretation or logical form.