Today

What would it mean to use formal methods in linguistics?

- The language sciences seek to discover useful things about human language; for this course, the things we care about are how human beings produce and understand language (as opposed to the related goals of e.g. making computers better, or helping students learn a second language better)

- You can describe this “how” at different levels of abstraction: you can look at what is literally going on in the brain; or you can look (as we usually do in linguistics) at a very abstract description of what is going on, which you could cast in terms of data structures, algorithms, etc., as versus the “implementation”

- In this course we will mostly talk about the language system in terms of what classes of functions/mappings the human language faculty must belong to, in order to account for the existing human languages

- We will try and develop fully formal treatments of small pieces of that mapping

An example

What does it mean to be formal?

Syntactic Structures (Chomsky 1957):

\begin{align}
S & \rightarrow NP + VP \\
NP & \rightarrow D + N \\
VP & \rightarrow V + NP \\
D & \rightarrow \text{the} \\
N & \rightarrow \text{man} \\
V & \rightarrow \text{hit} \\
V & \rightarrow \text{take} \\
V & \rightarrow \text{walk} \\
V & \rightarrow \text{read} \\
N & \rightarrow \text{ball}
\end{align}

All the things (like $NP$, $VP$, $\text{the}$, etc.) which are not separated by $+$ or $\rightarrow$ are individual symbols. Call any symbol that appears on the left side of an arrow a nonterminal symbol; call any symbol that appears only on the right side of an arrow a terminal symbol. Conventionally you would be given an intuitive description of how to use these rules as a “device for producing sentences of the language.” Here is an explicit version:
Algorithm for using (2) to produce a sentence of the language:

1. Let $d := S$
2. While $d$ contains at least one non-terminal:
   (a) Replace one of the non-terminals in $d$ with the string found on the right-hand side of an arrow which has that non-terminal on the left-hand side of it

Here we have the beginnings of a specification of English as a *formal language*. So, what do we mean by *formal*? Often the term “formal” suggests that something is precise, detailed, thorough, and so on. This is not precisely what is meant by “formal” in the sense we are using it here; here we are using it in the sense of “purely formal,” meaning, “relating to the form and nothing else.” What makes the system we just constructed “purely formal” is that we can generate English sentences (or whatever approximation to English sentences) simply by manipulating the symbols that are written on the page; a robot could do it without any knowledge of any other facts about those symbols, by repeatedly applying a few explicit and well-defined rules. That is why the device we just saw (a very small context-free grammar) is an example of a formal grammar: we can do everything we need to do with the grammar by playing a symbol-manipulation game.

**Formal languages and how to specify them**

The system we saw above for generating sentences was a simple context-free grammar (CFG). A CFG is a type of formal grammar. A formal grammar is:

1. A set of allowable symbols, $\Sigma$, called the alphabet
2. A set of rules, $R$, which specify what the well-formed sentences are

“Sentence” simply means “string of symbols.” “Sentence” here is a technical term. We do not need to be dealing with actual natural language sentences. We could be dealing with a programming language, or we could be trying to give a grammar for possible words, in which case a “sentence” of the formal language actually correspond to a word of natural language.

A formal language is simply some set of allowable strings of symbols (not necessarily a finite set). A formal grammar is one means of precisely specifying a formal language. Again, “language” is a technical term—but we will try as best as we can to have the “languages” we work with be meaningful abstract versions of real human languages. We call a “language” in this sense a “formal language” not because there is anything obviously more form-related about it than any other kind of language, but simply to mean “the sense of the term language that we use in the context of formal grammars”—i.e., a set of strings.

We usually do not specify the rules of a formal grammar as a single set $R$. In the case of a CFG, we usually specify it as:

1. An alphabet of terminal symbols, $\Sigma$, which are (all) the symbols that can appear in strings
2. A set of nonterminal symbols, $N$, disjoint from $\Sigma$ (and which therefore cannot appear in strings)
3. A special start symbol $S \in N$
4. A set of production rules, $P$ (the kind of rules we showed above)
5. A relation $\Rightarrow$ which tells us how to apply the rules. Since there is a standard method of applying production rules that works for any grammar like a CFG (essentially, “find what is on the left-hand side of a rule somewhere and replace it with what is on the right-hand side”), we never actually bother to specify $\Rightarrow$, but it is an integral part of the operation of the grammar!
A formal grammar consisting of rewrite rules is not the only way to specify a formal language, however. We will spend most of this course working with a different type of specification: using an automaton, or, in particular, a decider. One way of characterizing any automaton is as a specially restricted version of the most general kind of automaton, a Turing machine. Imagine this as an infinitely long tape and a head which can read from or write to that tape, and move along the tape, following some set of rules. A Turing machine is usually specified as:

1. An alphabet, $\Sigma$, which are (all) the symbols that can appear in strings given as input to the Turing machine
2. A set of states, $Q$, of which one state, $q_0$, is specially designated as the start state
3. $F \subseteq Q$, a set of accepting states
4. A second alphabet, $T \supseteq \Sigma \cup \{\bot\}$, called the tape alphabet: the set of all symbols that can occur on the tape, where $\bot$ is a special “blank” symbol
5. A transition function, $\delta : Q \times T \rightarrow Q \times T \times \{\text{left}, \text{right}\}$

The idea is that the Turing machine looks at the tape and which state it’s in, and, depending on these things (and only these things), $\delta$ will tell the machine what state to enter, what to write on the current tape sector, and which direction to move the head in. When we reach an accepting state, we look at the contents of the tape, and this is the output.

Automata (restricted kinds of Turing machines) can be used just like grammars, to characterize languages. If the machine finishes in an accepting state, then we say that the input string is in the language; otherwise, it is not in the language. Of course, we can also look at the output, and we could treat the output as the output of some function if we liked. Typically, however, we use these machines like this just to tell whether something is in a set or not, in which case we call them deciders or acceptors. A special kind of acceptor in which we only get to move to the right and we are not allowed to change anything on the tape is called a finite-state automaton. (Intuitively, this prevents the machine from having any kind of memory.) We will look more closely at finite-state automata next class.

Preliminary goal of this course

The first goal of this course is to show you how a few ways in which researchers have formalized parts of natural language grammars, that is, ways in which they have given a precise specification of a set of symbols, plus some system, either grammar or automaton, that specifies a set of strings that closely matches some interesting part of a particular natural language, such as the possible sentences of Swiss German or the possible words of English.

Reasoning about formal grammars

Syntactic Structures (Chomsky 1957):

The fundamental aim in the linguistic analysis of a language $L$ is to separate the grammatical sequences which are the sentences of $L$ from the ungrammatical sequences which are not sentences of $L$ and to study the structure of the grammatical sequences. . . . One way to test the adequacy of a grammar proposed for $L$ is to determine whether or not the sequences that it generates are actually grammatical, i.e., acceptable to a native speaker, etc. . . . A certain number of clear cases . . . will provide us with a criterion of adequacy for any particular grammar. For a single language, taken in isolation, this provides only a weak test of adequacy, since many different grammars may handle the clear cases properly. This can be generalized to a very strong condition, however, if we insist that the clear cases be handled
properly for each language by grammars all of which are constructed by the same method. . . . We then have a very strong test of adequacy for a linguistic theory that attempts to give a general explanation for the notion “grammatical sentence” in terms of “observed sentence,” and for the set of grammars constructed in accordance with such a theory.

The main goal of linguistics is not to construct grammars for individual languages. A linguistic theory is not supposed to tell us what the grammatical sentences are in a particular language. Rather, the main goal of linguistics is to find out how human beings construct mental grammars for individual languages. A linguistic theory is a theory of the class of possible mental grammars (not possible strings).

We will model mental grammars as formal grammars—actually, more often, as automata. We then want to classify those automata in some way by proving things about them. Here are some examples of things we often want to prove:

- The set of all languages that can be characterized by automata from class X are also languages that can be characterized by automata from class Y (i.e. all type X-languages are also type Y-languages; Y-automata are at least as “powerful” as X-automata)

- The set of all languages that can be characterized by automata from class X are also languages that can be characterized by automata from class Y, and conversely (i.e. the set of X-languages is equal to the set of Y-languages; X-automata and Y-automata are weakly equivalent, or string-equivalent)

- The union/intersection/concatenation/… of a language that can be characterized by an X-automaton with another language that can be characterized by an X-automaton can/cannot also be characterized by an X-automaton

- Some particular language that we can describe in words cannot be characterized by an X-automaton

We will see all of these as we go along. However, the most important ones are the last. With these kinds of proofs in hand, we can search for empirical data and state the boundary conditions are on the correct theory of how mental grammars work. Here are the two crucial cases we will work towards:

1. If we look at human languages as sets of strings of words or morphemes, then we will find that, in general, we cannot characterize what is syntactically acceptable in a human language using a finite-state automaton, or even using a context free grammar

   but

2. If we look at human languages as sets of strings of phonetic segments, then we will find that we can always characterize what is phonologically acceptable in a human language using a finite-state automaton, or something less powerful.

(9) Main goal of this course

The second and main goal of this course is to show how we can make (and prove) general statements about what class(es) of formal grammars and automata characterize the human language faculty.

About this course

This is the first time a course like this has run here (maybe anywhere). It is not a regular seminar course, because it has exercises. This is because half of what you learn when you use formal techniques is how, not what.
Some of the content of the course is standard for an upper-level course in formal language and automata theory, and, as such, the first homefun is mostly practice with standard proof techniques that are commonly needed in such a course (proof by induction, proof by contradiction, finding a counterexample). Apart from the first part of Wednesday’s lecture, it presupposes only some basic arithmetic and an understanding of these proof techniques. The readings for these proof techniques are posted on the website under today’s date; I will not go over them in class. Come to office hours this week so that we can discuss any issues.

The remainder of the content of the course will be a mix between phonology and formal language theory content; these two parts should reinforce each other. However, if anything in the course does not fit together as well as it should, just ask for a better explanation—after all, that is why you are here!

References