Computational Morphology

Lane Schwartz

University of Illinois at Urbana-Champaign

Week 05 of 16 — Day 08 of 29

Barbara Partee, Alice Ter Meulen, and Robert E. Wall. *Basic Concepts*: Chapter 16 of *Mathematical Methods in Linguistics*. 1993.

- Languages, grammars, and automata
- a natural language is simply a set of strings

- Not every possible sequence is in the language: we distinguish the grammatical strings from those that are ungrammatical.

- A grammar is some explicit device for selecting a subset of strings, those that are grammatical, from the set of all possible strings formed from an initially given alphabet or vocabulary.

- two classes of formal devices function as grammars:
- (1) automata, which are abstract computing machines
- (2) string rewriting systems

- Given a finite set A, a string is a finite sequence of occurrences of elements from A
- For example, if $A = \{a, b, c\}$, then acbaab is a string on A

- Strings are by definition finite in length.
- The set from which strings are formed is often called the vocabulary or alphabet, and this too is always assumed to be finite
- We also recognize the (unique) string of length 0, the empty string, which we will denote e
- Two strings are identical if they have the same symbol occurrences in the same order

- An important binary operation on strings is concatenation
- A frequently encountered unary operation on strings is reversal

- Concatenation is associative but it is not commutative
- The empty string is the identity element for concatenation

- Given a string x, a substring of x is any string formed from contiguous occurrences of symbols in x taken in the same order in which they occur in x.
- An initial substring is called a prefix, and a final substring, a suffix.

- We may now define a language (over a vocabulary A) as any subset of A.

- The study of formal languages is essentially the investigation of a scale of complexity in this patterning in strings.

- A formal grammar is essentially a deductive system of axioms and rules of inference which generates the sentences of a language as its theorems.

- a grammar contains just one axiom, the string consisting of the initial symbol (usually S),
- and a finite number of rules of the form $\psi \rightarrow \omega$, where ψ and ω strings, and the interpretation of a rule is the following: whenever ψ occurs as a substring of any given string, that occurrence may be replaced by ω to yield a new string .

- Grammars use two alphabets: a terminal alphabet and a non-terminal alphabet, which are assumed to be disjoint.

- the sentences of the language, are strings over the terminal alphabet
- intermediate strings in derivations may contain symbols from both alphabets.

- the sentences of the language, are strings over the terminal alphabet
- intermediate strings in derivations may contain symbols from both alphabets.

- V_T (the terminal alphabet) $= a, b$
- V_N (the non-terminal alphabet) $= \{S, A, B\}$
- S (the initial symbol — a member of V_N)
- R (the set of rules) $= S \rightarrow ABS$, $S \rightarrow e$, $AB \rightarrow BA$, $BA \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$

- A common notational convention is to use lower case letters for the terminal alphabet and upper case letters for the non-terminal alphabet.

- A derivation of the string abba by this grammar could proceed as follows:
- $S \Longrightarrow ABS \Longrightarrow ABABS \Longrightarrow ABAB \Longrightarrow ABBA \Longrightarrow ABbA \Longrightarrow aBbA \Longrightarrow abbA \Longrightarrow abba$
- the symbol $\Longrightarrow$ means "yields in one rule application"
- The sequence is said to be a derivation of abba, and the string abba is said to be generated by the grammar

- The language generated by the grammar is the set of all strings generated.

- Let $\Sigma = V_T \cup V_N$
- A formal grammar G is a quadruple (V_T, V_N, S, R), where V_T and V_N are finite disjoint sets, S is a distinguished member of V_T, and R is a finite set of ordered pairs in $\Sigma^* V_N \Sigma^* \times \Sigma^*$.

- Given a grammar $G = (V_T, V_N, S, R)$, a derivation is a sequence of strings $x_1, x_2, \ldots, x_n$ (n 1) such that $x_1 = S$ and for each $x_i (2 \leq i \leq n)$, x_i is obtained from x_{i-1} by one application of some rule in R.

- A grammar G generates a string $x \in V_T^*$ if there is a derivation $x1, \ldots, x_n$ by G such that $x_n = x$.

- The language generated by a grammar G, denoted $L(G)$, is the set of all strings generated by G.

- When the rules of a grammar are restricted to rewriting only a single non-terminal symbol, it is possible to construe grammars as generating constituent structure trees rather than simply strings.
- a tree diagram is ordinarily drawn upside down since the root is at the top and the leaves are at the bottom
- a tree is invariably singly rooted

- $A \rightarrow \psi / \alpha \_\_\_ \beta$
- the / is read "in the context", and where _ marks the position of the A

- Grammars and Trees
- $A \rightarrow \psi / \alpha \_\_ \beta$
- the / is read "in the context", and where _ marks the position of the A
- The interpretation of such a rule is that the symbol A can be replaced by the string ψ in a derivation only when the string α immediately precedes A and the string β immediately follows A

- Such rules are called **context sensitive** in contrast to rules of the form $A \rightarrow \psi$, which are called context free
- A context free rule, thus, is a context sensitive rule in which the context is null

- At the top are the most general grammars
- There are no restrictions on the form of the rules except that the left side cannot be the empty string.
- Chomsky dubbed such grammars 'Type 0,' and they are also sometimes called unrestricted rewriting systems

- Type 1: each rule is of the form $\alpha A \beta \rightarrow \alpha \psi \beta$ where $\psi \neq e$
- Type 2: each rule is of the form $A \rightarrow \psi$
- Type 3: each rule is of the form $A \rightarrow xB$ or $A \rightarrow x$

- In the above α, β, and ψ are arbitrary strings (possibly empty unless otherwise specified) over the union of the terminal and non-terminal alphabets; A and B are non-terminals, and x is a string of terminal symbols

- Type 1 grammars are also called context sensitive
- Type 2 grammars are called context free
- Type 3 grammars are called regular

- these classes of grammars do not form a strict hierarchy in the sense that each type is a subclass of the one with the next lower number
- because rules of the form $A \rightarrow e$ are allowed in Type 2 grammars, these are not properly contained in Type 1.

- the Type 3 languages are properly included in the Type 2 languages;
- the Type 2 languages not containing the empty string are properly included in the Type 1 languages;
- the Type 1 languages are properly included in the Type 0 languages.

- Languages and automata
- languages can also be characterized by abstract computing devices called automata
- An automaton is an idealized abstract computing machine

- Central to the notion of the structure of an automaton is the concept of a state.
- An automaton may also have a memory.
- For the simplest automata, the memory consists simply of the states themselves
- More powerful automata may be outfitted with additional devices, generally "tapes" on which the machine can read and write symbols and do "scratch work"

- Automata may be regarded as devices for computing functions, i.e., for pairing inputs with outputs, but we will normally view them as acceptors, i.e., devices which, when given an input, either accept or reject it after some finite amount of computation. In particular, if the input is a string over some alphabet A, then an automaton can be thought of as the acceptor of some language over A and the rejector of its complement. As we will see, it is also possible to regard automata as generators of strings and languages in a manner similar to grammars