Chapter 1

Learning Patterns in Sequences: Introduction

1.1 Strings and stringsets

A string is a finite sequence of symbols from some set of symbols Σ. The set of all possible strings is often noted Σ^*. The asterisk is a symbol due to Kleene, who is one of the great computer scientists of the twentieth century. It is often called the 'Kleene star.' Generally, the presence of the Kleene star on a set denotes the **free monoid** of a set, which is the set of all finite sequences of length zero or more from that set. The unique string of length zero is often denoted λ or ϵ.

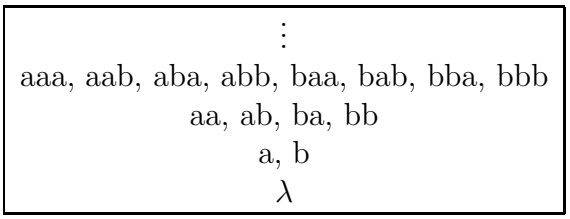


Figure 1.1: Strings of increasing length with $\Sigma = \{a, b\}$.

Here are some examples of sets of strings, also called **formal languages**, or **stringsets**.

1. Let $\Sigma = \{$`a, b, c, ..., z, .`$\}$. Then there is a subset of Σ^* which includes all and only the grammatical sentences of English (modulo capitalization).

2. Let $\Sigma = \{$`Advance-1cm, Turn-R-5°`$\}$. Then there is a subset of Σ^* which includes all and only the ways to get from point A to point B.

Exercise 1. Provide some more examples of stringsets relevant to linguistics.

1.2 The membership problem

The **membership problem** is the problem of deciding whether a string belongs to a set. The problem can be stated thusly: Given a set of strings S and *any* string $s \in \Sigma^*$, output whether $s \in S$. Is there an algorithm that solves this problem for a given S?

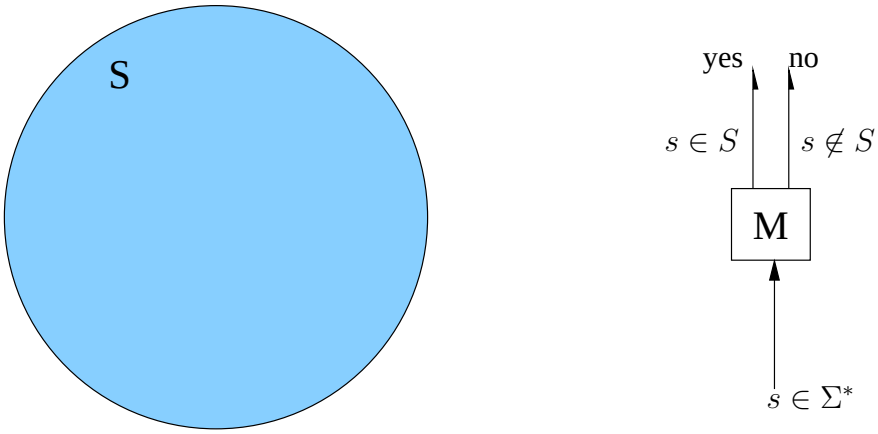
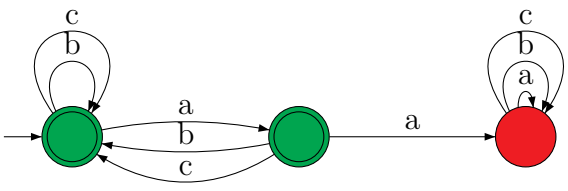


Figure 1.2: The membership problem

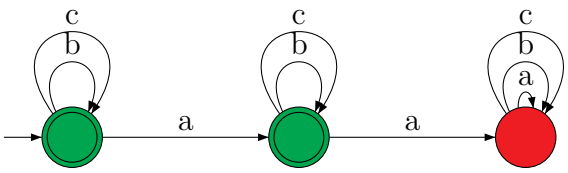
Example 1. A string belongs to S if it does not contain `aa` as a substring.

$s \in S$	$s \notin S$
abba	baab
abccba	aaccbb
babababa	ccaaccaacc
...	...



Example 2. A string belongs to S if it does not contain `aa` as a subsequence.

$s \in S$	$s \notin S$
cabb	baab
babccbc	babccba
bbbbbb	bbacccccccccacccc
...	...



Exercise 2. These finite-state machines are not the only algorithmic solutions to these membership problems? Provide other algorithms which solve these two membership problems.

1.3 Learning problems

There are many ways to define the problem of learning a stringset. Here are two informal ones just to get started.

1. For any set S from some given collection of sets: Drawing finitely many examples from S, output a program solving the membership problem for S.

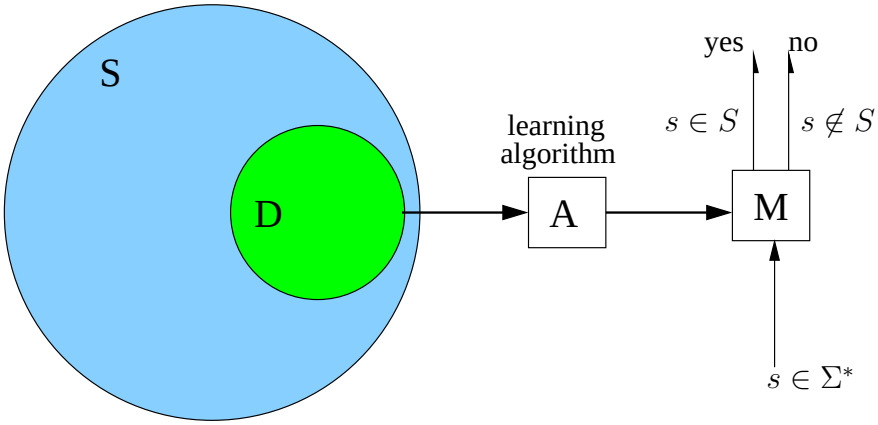


Figure 1.3: A Learning Problem with Only Positive Evidence

2. For any set S from some given collection of sets: Draw finitely many strings labeled as to whether they belong to S or not, output a program solving the membership problem for S.

These definitions are too informal. What does it mean to draw finitely many examples? Do I want the algorithm to succeed for any finite sample of strings providing information about the language? Why or why not?

Exercise 3. Improve the above definitions by making clearer what 'drawing' and 'output' mean. Try to write it formally if you can.

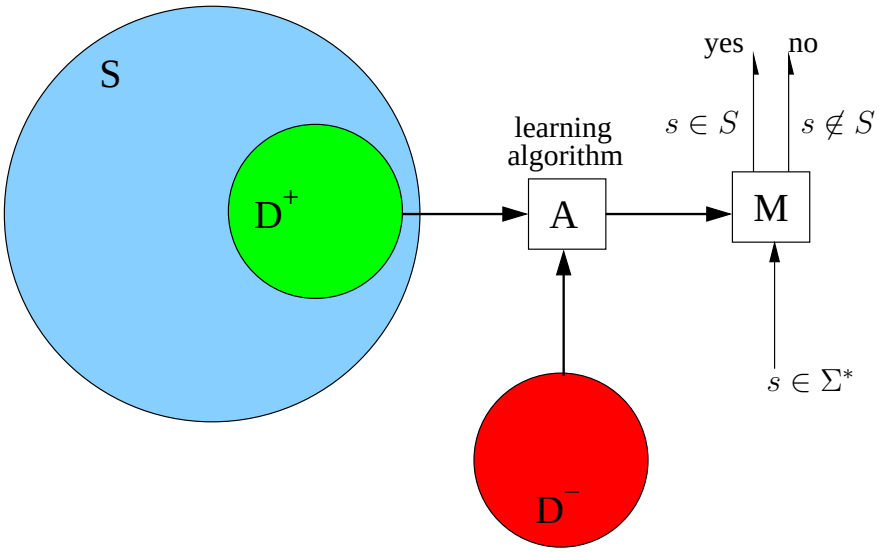


Figure 1.4: Learning Problem with Positive and Negative Evidence

Exercise 4. Recall Osherson *et al.* (1986):

"Of special interest are the circumstances under which these hypotheses stabilize to an accurate representation of the environment from which the evidence is drawn. Such stability and accuracy are conceived as the hallmarks of learning. Within learning theory, the concepts 'evidence,' 'stabilization,' 'accuracy,' and so on, give way to precise definitions."

How do your improved definitions address these concepts?

1.4 Generalizing a little bit

The functional perspective lets us generalize the foregoing a little bit. From a functional perspective a stringset S is associated with a function $f : \Sigma^* \rightarrow \{0, 1\}$. But we may be interested in other types of functions which have Σ^* for a domain.

function	Notes
$f : \Sigma^* \rightarrow \{0, 1\}$	Binary classification
$f : \Sigma^* \rightarrow \mathbb{N}$	Maps strings to numbers
$f : \Sigma^* \rightarrow [0, 1]$	Maps strings to real values
$f : \Sigma^* \rightarrow \Delta^*$	Maps strings to strings
$f : \Sigma^* \rightarrow \wp(\Delta^*)$	Maps strings to sets of strings

Exercise 5. Provide some specific examples of functions like the ones above relevant to linguistics.

Exercise 6. Try to write another definition of learning. It can be for any of the types of functions shown.

1.5 Classifying membership problems

The Chomsky Hierarchy provides one classification of membership problems (Chomsky, 1956; Hopcroft and Ullman, 1979).

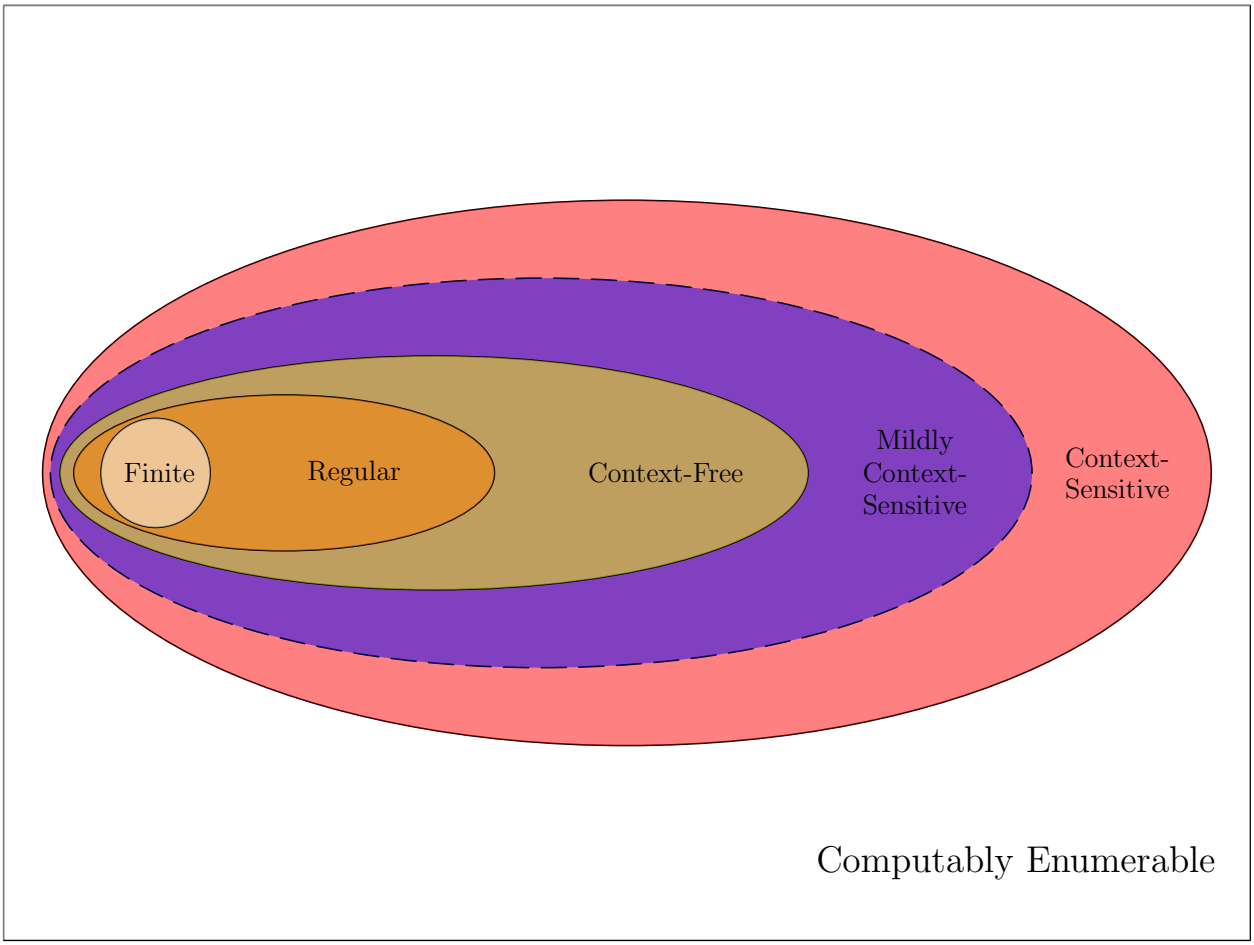


Figure 1.5: The Chomsky Hierarchy

Of particular interest is the class of regular languages, which may be defined as the class of stringsets whose membership problem is solvable with a finite-state acceptor. As you may know, zooming in on the class of regular languages reveals some more structure (McNaughton and Papert, 1971; Thomas, 1982).

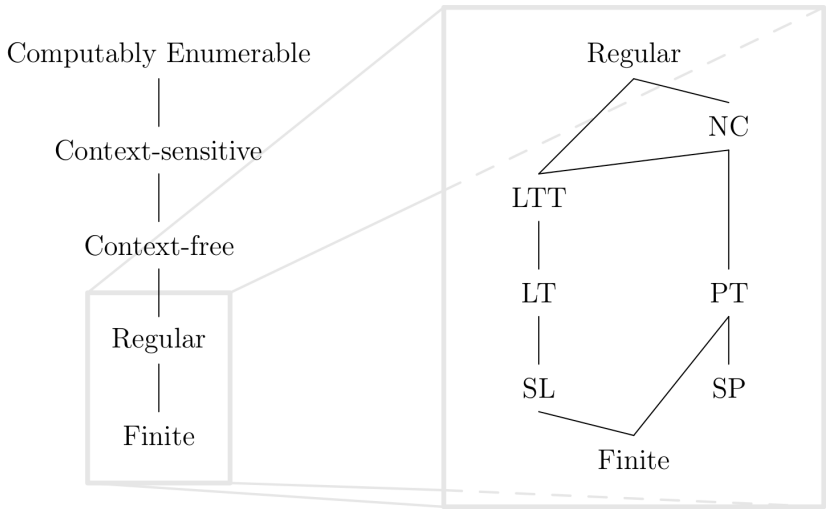


Figure 1.6: Room at the bottom.

These subregular classes are presented below from a model-theoretic perspective (Rogers *et al.*, 2010, 2013).

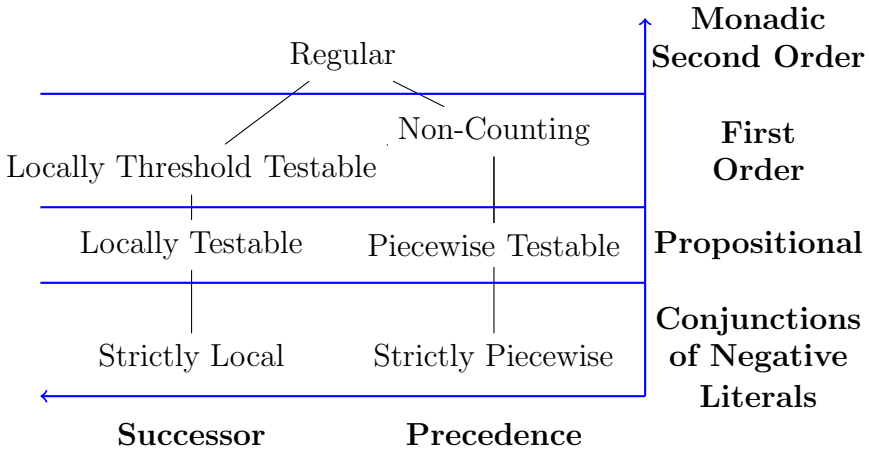


Figure 1.7: Subregular Hierarchies.

Bibliography

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 113124. IT-2.

Hopcroft, John E., and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

Osherson, Daniel, Scott Weinstein, and Michael Stob. 1986. *Systems that Learn*. Cambridge, MA: MIT Press.

Rogers, James, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In *The Mathematics of Language*, edited by Christian Ebert, Gerhard Jäger, and Jens Michaelis, vol. 6149 of *Lecture Notes in Artifical Intelligence*, 255–265. Springer.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar*, edited by Glyn Morrill and Mark-Jan Nederhof, vol. 8036 of *Lecture Notes in Computer Science*, 90–108. Springer.

Thomas, Wolfgang. 1982. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences* 25:370–376.