1. Regular languages as languages generated by FSA

When we did distributional analysis, we saw that linguistic units in natural language (roughly words) can be classified into grammatical categories or distributional classes:

<table>
<thead>
<tr>
<th>Grammatical category</th>
<th>Lexical Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper nouns</td>
<td>Joan, Bill, Philadelphia</td>
</tr>
<tr>
<td>Pronouns</td>
<td>He, she, it, they</td>
</tr>
<tr>
<td>Nouns</td>
<td>Dog, cat, candy</td>
</tr>
<tr>
<td>Intransitive verbs</td>
<td>Walk, sleep, snore</td>
</tr>
<tr>
<td>Transitive verbs</td>
<td>See, find, hug</td>
</tr>
<tr>
<td>Auxiliary verbs</td>
<td>Will, would, could, must</td>
</tr>
<tr>
<td>determiners</td>
<td>The, a, some, every</td>
</tr>
<tr>
<td>prepositions</td>
<td>With, in, on, to, before</td>
</tr>
<tr>
<td>adjectives</td>
<td>Tall, short, green</td>
</tr>
<tr>
<td>adverbs</td>
<td>Quickly, carefully, very</td>
</tr>
<tr>
<td>complementizers</td>
<td>That, if, whether</td>
</tr>
<tr>
<td>conjunctions</td>
<td>And, or, but</td>
</tr>
</tbody>
</table>

When people learn a language, they do not learn which word follows which other word. Rather, they learn which grammatical category can follow which other category. In this respect, learning a language can –perhaps – be viewed as learning of constructing the FSA that recognizes it. E.g., learning the language in (1) can be viewed as building the non-deterministic FSA below.

\[
L = \text{Det} \quad \text{Adj} \quad \text{Noun} \quad \text{TrVerb} \quad \text{Noun}
\]

a. Every dog eats mice
b. The sad tiger saw dogs
c. Some short black cat hears mice
2. Regular languages as languages generated by Right Linear Grammars

Regular languages are also generated by a rule formalism called Right (or Left) Linear Grammar (also called type 3 Grammar): A -> aB

1. A is a single symbol (corresponding to a state) called a ‘non-terminal symbol’
2. a corresponds to a lexical item
3. B is a single non-terminal symbol.

Formal definition of Right Linear Grammars
A right linear grammar is a 4-tuple <T, N, S, R>, where:
1. N is a finite set of non-terminals
2. T is a finite set of terminals, including the empty string
3. S is the start symbol
4. R is a finite set of rewriting rules of the form A-> xB or A-> x, where A and B stand for non-terminals and x stands for a terminal.

Formal example:
G1 = <T, N, S, R>, where T = {a, b}, N = {S, A, B}, and

\[ R = \begin{cases} 
  S \rightarrow aA \\
  A \rightarrow aA \\
  A \rightarrow bB \\
  B \rightarrow bB \\
  B \rightarrow b 
\end{cases} \]

Right Linear Grammar for Language L and its non-deterministic FSA:
3. Finite State Automata and Right Linear Grammar

Every FSA has a corresponding right linear grammar and vice versa. That is, for every FSA, there is an equivalent right linear grammar that accepts the same language, and vice versa.

Converting a right linear grammar to an equivalent FSA

1. S is the start state
2. Associate with each rule of the form A -> xB a transition in a FSA from state A to state B reading x
3. Associate each rule of the form A -> x with a transition from state A reading x to a final state, F.

Example: Representing G1 from previous section as FSA:

S -> detB
S -> detC
B -> adj B
B -> adjC
C -> nounD
D -> transitiveVerbE
E -> noun
Converting a FSA to a right linear grammar

1. The states of the FSA become non-terminals of the grammar, and the alphabet of the FSA become terminals of the grammar.
2. q0 becomes the start symbol S.
3. For each transition $\delta(q_i, x) = q_j$, we put in the grammar a rule $q_i \rightarrow xq_j$.
4. For each transition $\delta(q_i, x) = q_j$, where $q_j$ is a final state, we add to the grammar the rule $q_i \rightarrow x$.
5. If the start state q0 is also an accept state, imagine that you have a non-deterministic FSA, where $\delta(q_0, e) = q_0$

Example:

$$G_2 = <T, N, q_0, R>, \text{ where } T = \{a, b\}, \ N = \{q_0, q_1\}, \text{ and }$$

$$R = \begin{cases} 
q_0 \rightarrow aq_0 \\
q_0 \rightarrow bq_1 \\
q_1 \rightarrow aq_1 \\
q_1 \rightarrow bq_0 \\
q_0 \rightarrow b \\
q_1 \rightarrow a
\end{cases}$$

Question: Design the non-deterministic FSA and the Right Linear Grammar that generates the following language. Classify very simply as an adverb, ignoring the fact that not all adverbs can occur in its position)
(2) \[ F = \text{Det} \ (\text{(very) Adj})^{*} \ Noun \quad \text{TrVerb} \quad \text{Noun} \]

a. The dog eats mice
b. The sad tiger saw dogs
c. The very sad tiger saw dogs
d. The short black cat hears mice
e. The short very black cat hears mice
f. The very short black cat hears mice
g. The very short very black cat hears mice

4. Is English a regular language?

The demonstration that English is not regular (i.e. that English is not recognized by a FSA nor can be generated by a Right Linear Grammar) was one of the first results achieved by the nascent field of linguistics in the 1950s. Here we will sketch one formal proof based in one type of dependency in English (Relative clause formation). We will also comment on a type of long distance dependency; you should check the sketch of the corresponding formal proof in Partee et al. p. 578ff.

Relative clause dependency:
1. Let us first consider the following language G:
   \[ G = \{ \text{The cat died, The cat the dog chased died, The cat the dog the rat bit chased died, The cat the dog the rat the elephant admired bit chased died} \ldots \} \]
   \((\text{the noun})^n \ (\text{transitive verb})^{n-1} \ \text{intransitive verb}\)
   This is comparable to the language \( \{0^n1^{n-1} | n \geq 0 \} \). We have already shown that a similar language – namely, \( \{0^n1^n | n \geq 0 \} \) – is not regular using the pumping lemma. The same procedure would show that G is not regular.

2. Let us now consider the language H:
   \[ H = (\text{the noun})^* (\text{transitive verb})^* \{\text{died}\} \]
   This language is regular.
   Question: Show that this language is regular.

3. G is the result of intersecting English (viewed as a set of strings) with the regular language H. Given that regular languages are closed under intersection, if English were regular, G would be also. But G is not regular. Hence, English is not regular.

Long distance dependency:
Finite state automata have no real memory. So, they cannot model sentences that contain items that depend on each other across arbitrarily long strings (Long distance dependency).

(3) Either the girl eats ice cream, or the girl eats candy
(4) If the girl eats ice cream, then the boy eats hot dogs
(5) *Either the girl eats ice cream, then the girl eats candy
(6) *If the girl eats ice cream, or the boy eats hot dogs
These long distance dependencies can be nested in another long distance dependencies.

(7) If either the girl eats ice cream or the girl eats candy, then the boy eats hot dogs.

(8) Either if the girl eats ice cream then the boy eats ice cream, or if the girl east ice cream, then the boy eats candy.

To handle all possible long distance dependencies, we would need an infinite number of states, which won’t fit inside a finite brain.

Extra exercises.
For each of the following languages, construct a FSA (deterministic or not) and its corresponding Right Linear Grammar:
(a) \{ab, bb\}
(b) \(a^*ab^*\)
(c) \{w: w contains at least one occurrence of a or at least one occurrence of b (or both)\}
(d) \{w: w contains at most three as\}