CSME 206A Natural language & Speech Processing **Spring Semester**

Lecture 3: Finite Automata and Morphological Parsing

Lecturer: K.R. Chowdhary *: Professor of CS*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the lecturer.*

3.1 Finite Automata

The finite automata are the machines which recognize regular languages, i.e, languages represented by regular expressions (RE). Hence, there is relation of implication like this, $FA \rightarrow REGLANG \rightarrow RExp \rightarrow FA$, in other words, it is a bijection between all three entities. Let us consider that, talk or sound of a sheep (sheep-talk language) can be represented by strings *baa*, *baaa*, *baaa*, etc., depending on the length of sound. This can be represented by a RE $baa^{+}!$, however, we will prefer it to be represented by /baa+!/, using a format of pronunciation. This sound can be recognized by a FA shown in the figure 3.1.

Formally, a FA is represented by $M = (Q, \Sigma, q_0, \delta, F)$, where $\delta : Q \times \Sigma \rightarrow Q$, $\Sigma = \{a, b, !\}$, $F = \{q_4\}$, $L(M) = \{baa!, baaa!, baaaa!, \dots\}$. The recognition process for the language string through FA, is represented by the algorithm 1. The FA's tape is divided into squares, called *index* positions, each position holding one symbol from alphabet Σ, for n positions, where $n = |w|$, and w is input string.

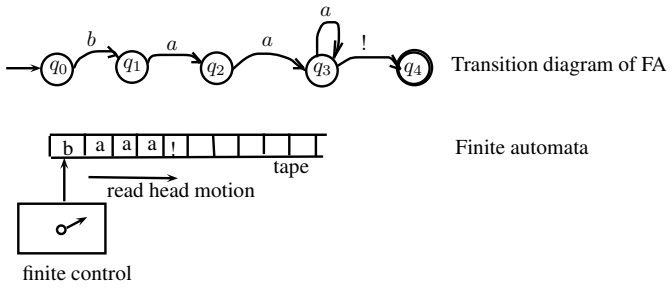

Figure 3.1: Finite Automata.

The recognition of a string is search of a tree. Considering a the recognition of some string of say length 5, for alphabet set $\Sigma = \{a, b\}$, one needs to perform a worst case search of $O(2^5)$ in space and time, when breadth-first search (BFS) is performed. In general, if size of alphabet $|\Sigma| = m$, and length of string (word) is h, then space and time both have complexities equal $O(h^n)$.

However, in case of depth-first search (DFS) case, space is $O(2 \times 5)$ (general case $O(hn)$), and time complexity remains the same (figure 3.2). Since, the ordinary brute force algorithms, like DFS and BFS are highly inefficient, better algorithms like best-free search, A^*, and simulated annealing are considered superior. However, due to combinatorial explosion of number of states generated, even shorter length strings, makes it difficult to efficiently process the input

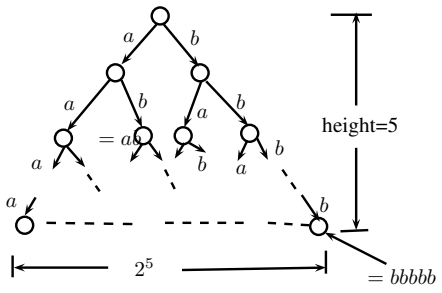

Figure 3.2: Search-tree.

Algorithm 1 : function dfa-recognize(tape, machine) return accept/reject;

```
1: index ← initialstate
2: while True do
3:   if end of input then
4:     if current state== accept state then
5:       return accept
6:     else
7:       return reject
8:     end if
9:   else
10:    if transition-table[current-state, tape[index]] == empty then
11:      return reject
12:    else
13:      current-state ← transition-table[current-state, tape[index]]
14:      index++
15:    end if
16:  end if
17: end while
```

for recognition.

3.2 English Language Morphology

Morphology is study of, how the words are constructed. Construction of English language words through attachment of prefixes and suffixes (both together called affix) are called *concatenative morphology*, because a word is composed of number of morphemes concatenated together. A word may have more than one affix, for example rewrites (re+write+s), unlikely (un+like+ly), etc. There are broadly two ways to form words using morphemes:

1. *Inflection:* Inflectional morphology form the words using the same group word stem, e.g., write+s, word+ed, etc. The table 3.1 shows the words constructed using inflective morphology.
2. *Derivation:* Derivations morphology produces a word of different stem, for example computerization (noun) from computerize (verb).

Table 3.1: Inflectional Morphology.

Type	Regular nouns	Irregular nouns
Singular	cat | thrush	mouse | ox
Plural	cats | thrushes	mice | oxen

The examples of *regular verbs* are walk, walks, walking, walked. Similarly, *irregularly inflected* verbs are eat, eats, eating, ate, eaten, catch, catches, cut, cuts, cutting, caught, etc.

The derivation is a combination of word stem with *grammatical morpheme*, usually resulting in a word of different class. For example, formation of nouns from verbs and adjectives. The table 3.2 shows the examples of derivational morphology.

3.3 Morphology and Finite-state Transducers

Table 3.2: Derivational Morphology.

Suffix	Base verb/adjective	Derived Noun
-action	computerize (V)	Computerization
-ee	appoint (V)	appointee
-er	kill (V)	killer
-ness	fuzzy (A)	fuzziness

To know the structure of a word we perform the *morphological parsing* for that word. Given a *surface form* (input form), e.g., "going" we might produce the parsed form: *verb-go + gerund-ing*. Morphological parsing can be done with the help of *finite-state transducer*. It is called transducer, because, unlike a finite automaton, which produces yes/no (accept/reject) output, the finite state transducer produces longer output, one output symbol for each input symbol.

A *morpheme* is a meaning bearing unit of any language. For example,

fox: has single morpheme, *fox* and,

cats: has two morphemes, *cat, -s*.

Similarly, eat, eats, eating, ate, eaten have different morphemes.

Some examples of mapping of some of certain words and corresponding morphemes are given in the table 3.3. The mapping of input and output correspond to the input and output of finite state machines.

In speech recognition, when a word has been identified, like cats, dogs, it becomes necessary to produce its morphological parsing, to find out its true meaning, in the form of its structure, as well to know how it is organized.

The features, like N (noun), V (verb), specify additional information about the word stem, e.g., $+N$ means that word is noun, $+SG$ means singular, etc.

We require following databases for building morphological parser:

1. **Lexicon:** list of stems, and affixes, plus additional information about them, like *+N, +V*.
2. **Morphotactics:** Rules about ordering of morphemes in a word, e.g. *-ed* follows a verb (e.g., worked, studied), *un* (undo) precede a verb, for example, unlock, untie, etc.
3. **Orthographic rules** (spelling): For combining morphemes, e.g., city+ *-s* gives cities and not citys.

Table 3.3: Morphemes.

Input Words	Morphological parsed output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +SG
gooses	goose +V +3SG
caught	catch +V +PAST

We can use the *lexicons* together with *morphotectics* (rules) to recognize the words with the help of finite automata in the form of stem+affix+part-of-speech (N, V, etc). The figure 3.3 shows the basic concept of morphological parsing of nouns. Recognition of nouns by FA is subject to reaching to final state (marked by double circle in figure) of FA. Table 3.4 shows some examples of regular and irregular nouns.

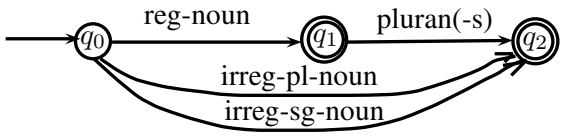

Figure 3.3: Morphological Parsing of nouns.

A similar arrangement is possible for *verb* morphological parsing (see figure 3.4, and table 3.5). The lexicon for verbal inflection have three stem classes (*reg-verb stem, irreg-verb stem,* and *irreg-past-verb*), with affix classes as: *-ed* for past and participle, *-ing* for continuous, and 3rd singular has *-s*.

Table 3.4: Regular and Irregular nouns.

Reg-noun	Plural	Irreg-noun	Irreg-sg-noun
fox	-es	goose	geese
cat	-s	sheep	sheep
dog	-s	mouse	mice

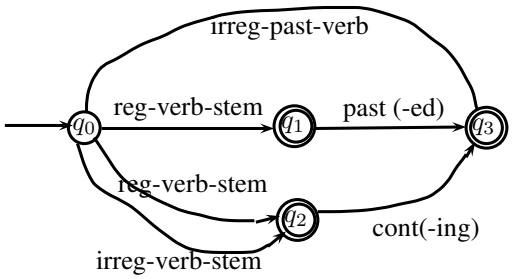

Figure 3.4: Morphological Parsing of verbs.

Adjectives can be parsed in the similar manner like, the nouns and verbs. Some of the adjectives of English language are: big, bigger, biggest, clean, cleaner, cleanest, happy, unhappy, happier, happiest, real, really, unreal, etc. The finite automata in figure 3.5 is showing the morphological parsing for adjective words.

Table 3.5: Regular and Irregular verbs.

Reg-verb	Past	Irreg-verb	Irreg-past-v	Cont.	3sg
walk	-ed	cut	caught	-ing	-s
fry	-ed	speak	ate	-ing	-s
talk	-ed	sing	eaten	-ing	-s

At the next stage, the lexicon can be expanded to sub-lexicons, i.e, individual letters, to be recognized by the finite automata. For example, regular-noun in figure 3.3 can be expanded to letters "*f o x*" connected by three states in a transition diagram, regular verb stem in figure 3.4 can be expanded by letters "*w a l k*", and so on, as shown in figure 3.6.

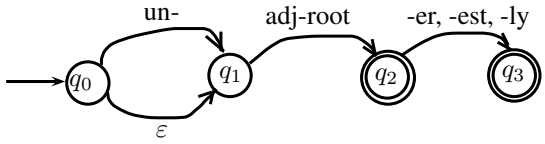

Figure 3.5: Morphological Parsing for adjectives.

Exercises

1. Write the algorithms for following:

(a) Parsing of regular nouns.

(b) Parsing of irregular nouns.

(c) Parsing of regular verbs.

(d) Parsing of irregular verbs.

(e) Parsing of adjectives.

2. What are the databases required for morphological parsing? Describe each, with its possible structure.

3. Perform the study of information available in any English dictionary, and list the information content in that, particularly that which is helpful for morphological parsing.

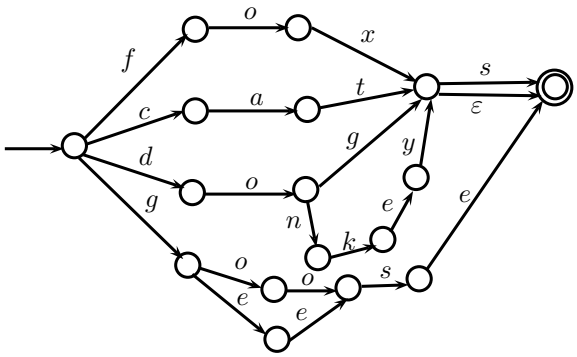

Figure 3.6: Morphological Parsing for noun words in details.

References

[1] D. Jurafsky and J. Martin, "Speech and Language Processing," *Pearson India*, 2002, Chapter 2, 3.