A Soundex Implementation in OCaml

Eray Özkural

February 7, 2017

I have implemented the Soundex algorithm using ocaml language. First, I have modified an old imperative directed graph implementation of mine so that it supports labels of any type and multi-graph instead of a simple graph. [1]

The transition function is represented by this graph, and it is exactly the same thing as the diagram of an FST. The only remaining variables are the starting state and the set of final states which are encapsulated in a record. The FST and other code has been implemented as separate modules which can be re-used easily. The transducer algorithm has been programmed as a simple recursive function which finds a matching prefix among the labels of adjacent edges of the given state in the graph of transition function, emits the output symbol, and calls itself recursively to construct a list of output symbols which are subsequently concatenated to obtain the output string.

The design of the soundex algorithm is achieved by cascading transducers designed for each step. The design of step a and b is straightforward but the other steps require a bit of explanation. In all of the transducers, I was able to express the FSTs at a very high level since ocaml has higher order functions such as map and iter for List and Array types. For instance in fstb, I have a line that is:

```
List.iter (fun x -> add d (1,1) (x,"1")) ["b";"f";"p";"v"];
```

These constructs reduce the code size considerably.

The design of step c is achieved as follows, for each digit we have a separate state $10 + i$ that keeps track of repetitions and does not emit anything as long as the same digit is read. Whenever the input comes to another digit or to end of input, the state's character is emitted.

In the design of d, we apply usual finite state methods. There is a state for each of the digits output. FST terminates after the state 4 corresponding to 3rd digit output. If end of string is seen before, padding zeroes are added appropriately.

To simplify implementation, character "#" is treated as sentinel for the input.

[1] Since the FST diagram has multiple edges among a pair of vertices