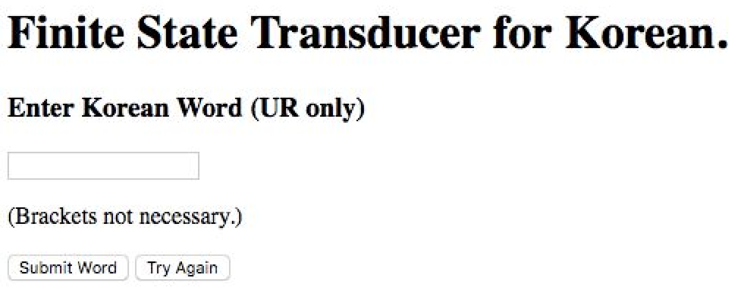
For the implementation of the finite state transducer, we create a simple web application to model it using HTML, Javascript, and the Canvas API. The web application allows the user to input a Korean word in UR form. The program takes the inputted word and applies our three rules: laryngeal neutralization, manner neutralization, and palatal neutralization. The following picture shows what it looks like when you first start the program:



Finite State Transducer for Korean.

Enter Korean Word (UR only)

(Brackets not necessary.)

Because our rules only care about the letter at word-final position, we only care about what the last letter is. The program takes the inputted word and reverses the order of the letters. A very important fact is that finite state transducers is only allowed to read one letter at a time from an input string. In programming, we call the input word as “input string”. Therefore, by reversing the order of the string in the beginning, we only care about what the first letter of the newly-reversed string. This means that we can simply send just the first letter of the new string to the finite state transducer, apply our rules if necessary, and then leave the rest of the letters alone.

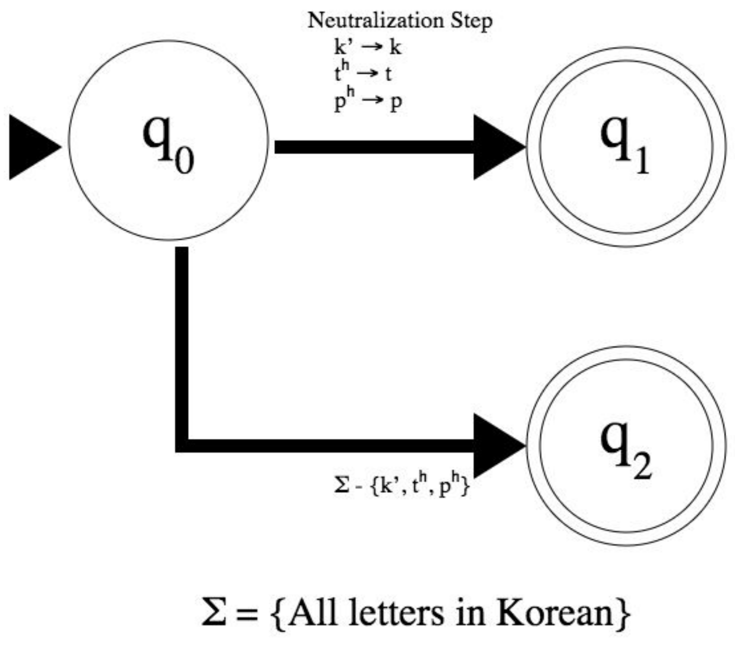
If the inputted word is fits the condition of the rule, meaning that the word’s environment matches what the rule is looking for, then the rule is applied and the program outputs the SR form of the Korean word. We do this with several if-statements in the code to catch for specific conditions.

After the user inputs a word, it outputs the SR form of the word, the rule that is applied, and the finite state transducer diagram for the applied rule. The following figure shows an example output of the program:

UR : /muk'/

SR : [muk]

Rule Applied : Laryngeal Neutralization



As we see from the example from this diagram, the user enters the Korean word /muk’/ in UR form. The program outputs the SR form [muk], outputs that the Laryngeal Neutralization is applied, and the finite state transducer itself. We trace through the finite state transducer, we begin with the state q_0 where the reverse order of user’s word is inputted. Each letter of the word takes its turn going through the transducer. The first letter that goes through is basically the last letter of the user’s word. At q_0, conditions are check for the letter. In this example, if the letter is a k’, t^h, or p^h, then it goes through the process of neutralization; the arrow leading to state q_1 represents the process the letter goes through. In this example, if the letter is not k’, t^h, or p^h, then it goes through the other arrow going into q_2, where Σ represents all possible IPA characters in Korean. Σ - {k’, t^h, or p^h} means the set of all possible Korean letters except k’, t^h, or p^h. Both q_1 and q_2 are end states, indicated by the double circles, meaning that it stops when those states are reached.

In other words, this diagram is represented in the programming language Javascript by using conditional statements checking the first letter of the reversed inputted word because we only care about the letter at word-final position. The web application uses HTML for the actual webpage and implementation of the buttons and input box. Lastly, the web application uses the Canvas API for drawing the finite state transducer diagram.

The web application works for any input the user enters. However, it is not implemented to catch whether the input word is an actual word in Korean. Also, the web application is limited to just the three Korean rules at word-final position. To run, download the .html file provided and double-click to begin.