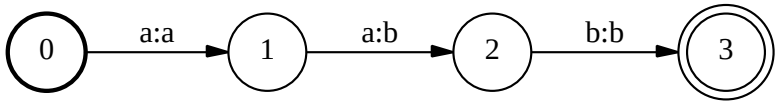
Finite-State Transducers

The following problems relate to *finite state transducers* a computational device or "machine" that is used a lot in computational linguistics.

A simple example is shown below:



1.fst

The machine has a finite set of *states* (hence the name "finite state") designated here with circles with numbers in them. There is a specified *initial state*; we will always note that state with the number "0". And there are one or more final states, which are conventionally notated with a double instead of single circle. And there may be states that are neither initial nor final. So in the machine above, there are four states, one initial state (0) and one final state (3).

Between the states there are *arcs*, and these arcs are labeled with pairs of symbols from an *alphabet*. An alphabet can be any finite set of symbols; here we will just use letters. The symbol on the left of the ":" on the arc is the *input* symbol, and the one on the right is the *output* symbol.

The machine is called a transducer because it *transduces* strings of symbols — e.g. a word — into other strings of symbols. How that works is as follows. Let us say you have a string *aab*, and imagine I have a little pointer that points to where I am currently in that string; initially it will point to the beginning letter. You start in the initial state of the machine, and you ask: given the letter where my little pointer is pointing, is there any arc that matches that first letter on its input label? In this case the answer is "yes": there is an arc labeled "a:a" that leaves that state, and goes to state 1. So I do three things:

- Output — from the arc's output label — the symbol "a"
- Move the machine from state 0 to state 1
- Move my little pointer to the second letter of the string (so it now points at the second a.

I then continue the process from state one and the second position of the string. In this case there is also an arc, labeled "a" on the input and "b" on the output, and my little input pointer is pointing at an "a", so I:

- Output — from the arc's output label — the symbol "b"
- Move the machine from state 1 to state 2
- Move my little pointer to the third letter of the string "b"

Now I'm looking at the "b" and I ask the same questions and by this point you should be able to see that I can:

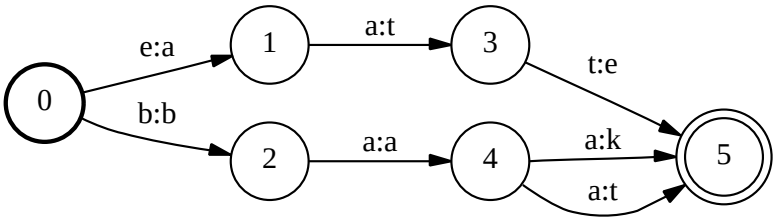
- Output — from the arc's output label — the symbol "b"
- Move the machine from state 2 to state 3
- Move my little pointer to the next position — which is the *end* of the string.

Now, notice I've reached the end of the string *and* I am in state 3, which is a final state. If, and only if, both conditions hold — I've used up my input and I am in a final state — then the machine has successfully read the input *and* successfully output a string. In this case it read *aab* and output *abb*.

If I had given it a string such as *abb* the machine above would fail to match the input, and hence would fail to give any output.

Problem 1: At which state will *abb* fail in 1.fst?

The machine we discussed is rather uninteresting since it only allows one input and output. But transducers can have more than one input and output. The following machine allows for a couple of input strings *eat* and *baa*, and three output strings:

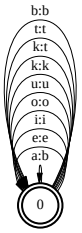


2.fst

Note that at state 4 there are *two* possible arcs to take on an input symbol *a*..

Problem 2: What are the output(s) for *eat*? What are the output(s) for *baa* in 2.fst?

Finite-state machines can have *loops* which are arcs or sequences of states and arcs that lead you back to a state that you've been to before. The following machine has nothing but loops: there is one state, which is both initial and final. You can read any string over the alphabet {a, e, i, o, u, k, t, b}. For example, you can read *aeeeeeeektttbiuiuiuiu* by simply reading a symbol, moving along the arc back to the initial state, and repeating the process. Once you've used up the string you will be in the initial state again, but since this is also a final state, the operation was successful.

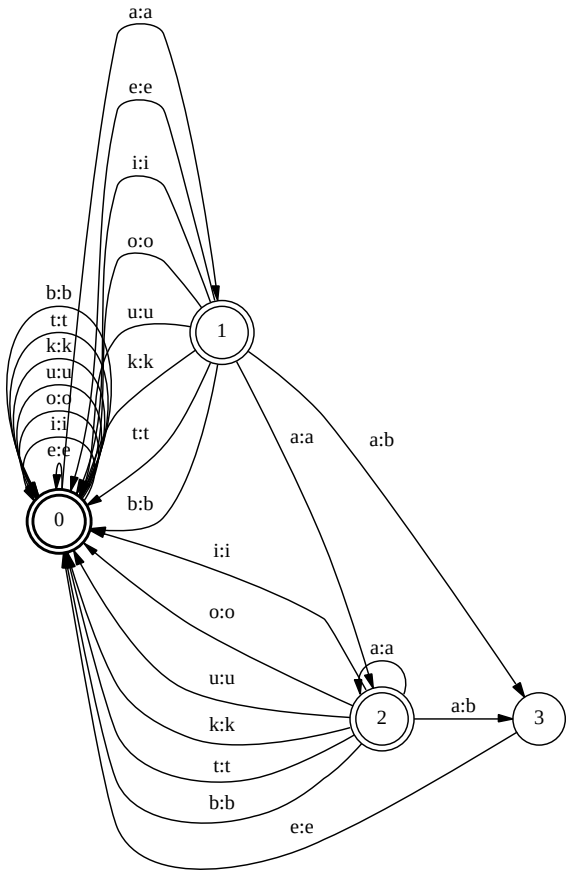


3.fst

For this machine nearly all of the arcs map symbols onto themselves, but there are some exceptions and some cases where a given input symbol might map to more than one possible output.

Problem 3: What are the output(s) for *kat* given the 3.fst? What are all the possible outputs for *kak*.

Here is a more complicated machine:



4.fst

See if you can figure out what it does:

Problem 4: In 4.fst, what are the output(s) for:

- **aabk**
- **aaek**
- **baek**
- **kaek**

Now that you are familiar with how FST's work, you are ready to help us solve a problem related to a simple cypher that was developed by the late Quinneas E. Dogil and found among his papers. Actually what was left was a sample of text and a transdsucer that implements the cypher. The transducer seems to have been run word by word on the input text, to produce examples like the text below.

The text is as follows and the cypher FST is on the final page:

lipg mtbgf anc mykfn symgs akb hzg lrhlfgs njbpxlh lighl hn qlas xinhanfnh a oyq orhabn xinvfakfc zn uarfghw anc wycavmhfc qi qlf yjbubsahabn qlmh acj iyn ajf xjfmhfc kgpmj

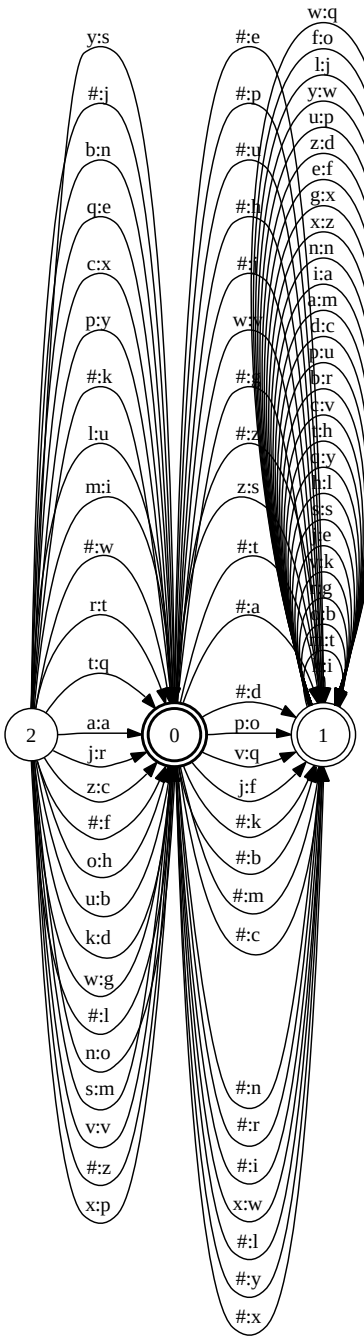
oiq gy ajf knxmxfc zn a jjfmh xakaj grg qyshanx glfhlfg qlmh orhabn hj anw orhabn mi xinvfakfc anc mi wycavmhfc xrn uinx kncpgf gy ajf iyh hn a jjfmh nrhhjfoafjc he qlmh grg gy frkf xitf qi wycavmhf a yighabn he qlmh lafjc ad a lanmj tyshanx ycmvf lig qlbsf glb fygf jrkf qlfag uakfs qlmh qlmh orhabn iaxlh uakf zb zd achbxfhlfg lahhanx anc yjbufg qlmh gy mlbpjc wi qlas

nzh zn a urgxfg mynsf gy xrn oih wycavmhf gy xrn oih xinsfvgmhf gy xrn oih frjjbq qlas jjbpnc qlf njmkf iyn uakanx anc wymc glb mbgpxxjfc fygf frkf xinsfvgmhfc zb lrg axbkf hzg yibg yiqfg qi amc hj wyhgmvh qlf gigjc gajj uahhjf oihf oig uinx tytftrfg glmh gy mrw fygf nzh zb xrn oykfg ligxfh glmh qlfw wac fygf zb zd lig bd qlf uakanx trhlfg qi ny wycavmhfc fygf qi qlf bnoanaslfc gigi glavl qlfw glb lipxlh fygf frkf qlps lrg mi oirjw amkmnvfc zb zd trhlfg lig bd qi ny fygf wycavmhfc qi qlf jjfmh qrsi tytmananx nyobgf bd qlmh ljbt qlfsf finbgfc wymc gy qrif znvgfmsfc

wykbhabn qi qlmh xrpsf lig glavl qlfw jrkf qlf ursh lzjj iymspgf he wykbhabn qlmh gy fygf faxljw tysbjkf qlmh qlfsf wymc mlmjj oih frkf wafc zn vran qlmh qlas orhabn bncfg jic mlmjj frkf a oyq naghl he ljffcbt anc qlmh jikfgntfnh he qlf yybujf nh qlf yybujf lig qlf yybujf mlmjj oih yygasl ljbt qlf krghl

Unfortunately the transducer as drawn was rather messed up, and there were a bunch of missing input arc labels, indicated in the figure with hash marks.

Problem 5: What is the text and and what are the missing input labels on the arcs?


2
0
1
a:a
j:r
z:c
#:f
o:h
u:b
k:d
w:g
#:l
n:o
s:m
v:v
#:z
x:p
t:q
r:t
#:w
m:i
l:u
#:k
p:y
c:x
q:e
b:n
#:j
y:s
#:d
p:o
v:q
j:f
#:k
#:b
#:m
#:c
#:n
#:r
#:i
x:w
#:l
#:y
#:x
#:t
#:a
z:s
#:z
#:x
w:x
#:h
#:u
#:p
#:e
i:t
i:i
w:q
f:o
l:j
y:w
u:p
z:d
e:f
g:x
x:z
n:n
i:a
a:m
d:c
p:u
p:r
c:v
c:h
c:y
s:l
s:s
s:e
s:k
s:g
s:b