ÍNDICE

☆SOME of the Available FST Operations (OpenFST Library)

– http://www.openfst.org

FST Transducers

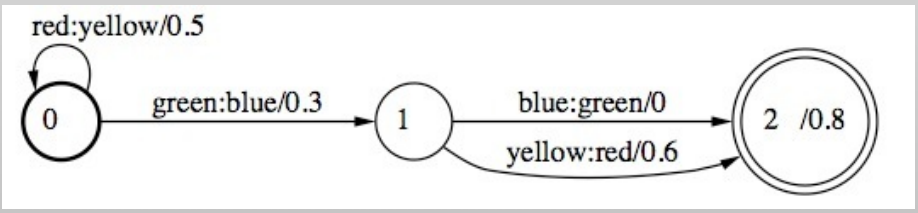
- *Definition of the symbols* (syms.txt)

```
red       1
green     2
blue      3
yellow    4
```

- *Definition of a transducer* (t.txt)

```
0   0   red     yellow    .5
0   1   green   blue      .3
1   2   blue    green
1   2   yellow  red       .6
2   .8
```

- *Graphical representation* (t.ps)



F S T T r a n s d u c e r s

- ***Definition of the symbols*** **(syms.txt)**

```
red       1
green     2
blue      3
yellow    4
```

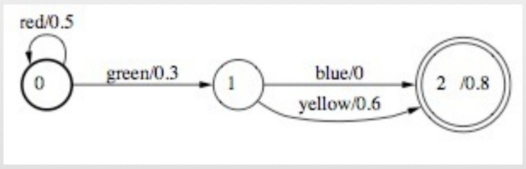
- ***Definition of a transducer*** **(t.txt)**

```
0    0    red      yellow     .5
0    1    green    blue       .3
1    2    blue     green
1    2    yellow   red        .6
2    .8
```

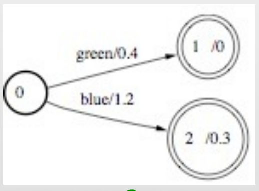
- ***Geração da versão binária***
  - **fstcompile** --isymbols=syms.txt --osymbols=syms.txt **t.txt | fstarcsort > t.fst**
- ***Geração da versão gráfica***
  - **fstdraw** --portrait --isymbols=syms.txt --osymbols=syms.txt **t.f | dot -Tpdf > t.pdf**

U N I O N O F T R A N S D U C E R S

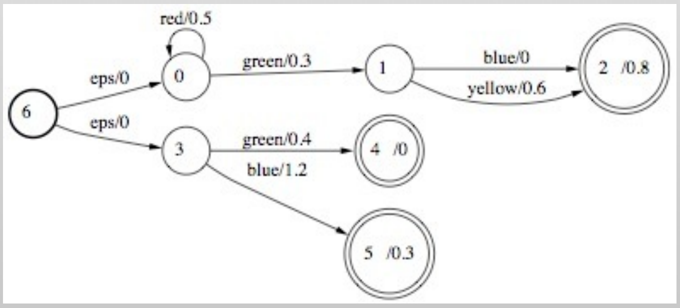
```
fstunion A.fst B.fst > C.fst
```



A.fst



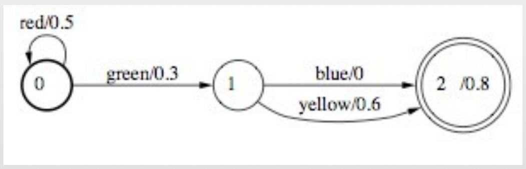
B.fst



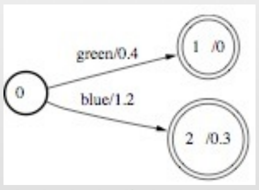
C.fst

CONCATENATION OF TRANSDUCERS

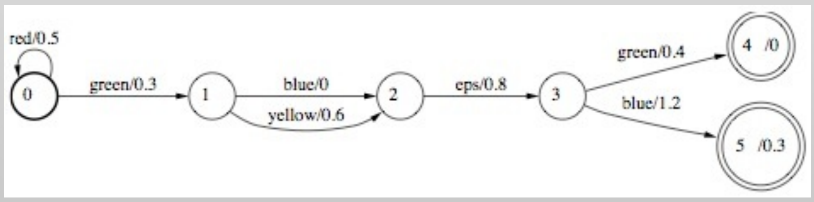
```
fstconcat A.fsm B.fsm > C.fsm
```



A.fst



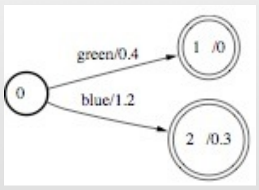
B.fst



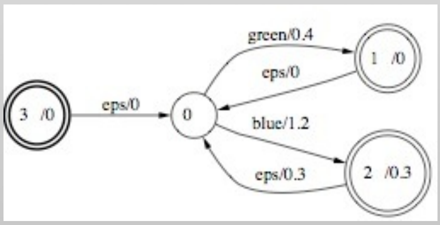
C.fst

CLOSURE OF TRANSDUCERS

```
fstclosure B.fst > C.fst
```



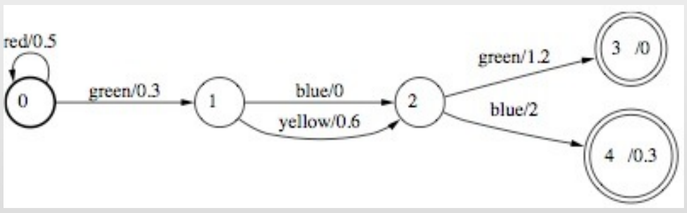
B.fst



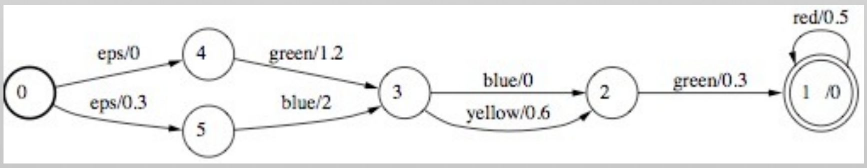
C.fst

“REVERSAL” OF TRANSDUCERS

```
fstreverse A.fst > C.fst
```



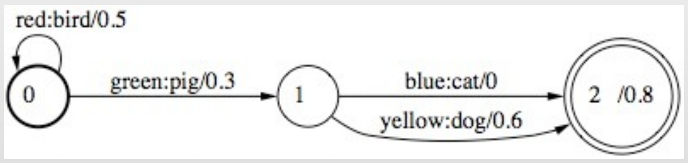
A.fst



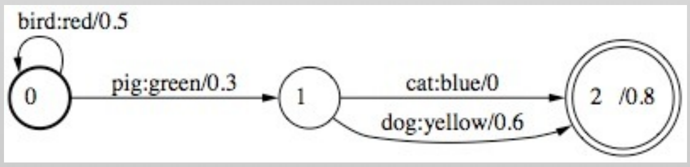
C.fst

INVERSION OF TRANSDUCERS

```
fstinvert A.fst > C.fst
```



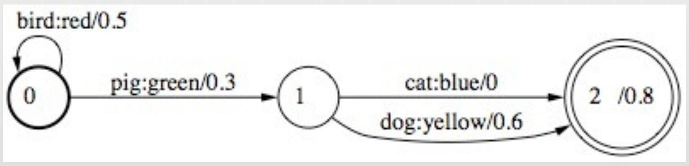
A.fst



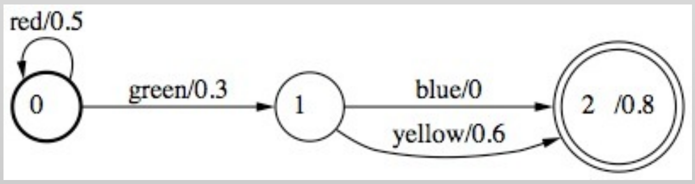
C.fst

PROJECTION OF TRANSDUCERS

```
fstproject --project_type=output  A.fst > C.fst
```



A.fst



C.fst

COMPOSITION OF TRANSDUCERS

- *To obtain the composition of two transducers:*
  - Creates a new state (x,y) for all the possible pairs $x \epsilon Q_1$ and $y \epsilon Q_2$
  - The transition function of the composition is defines by

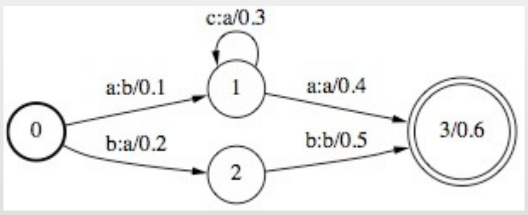
    $\delta((x,y),i:o)=(v,z)$

    if

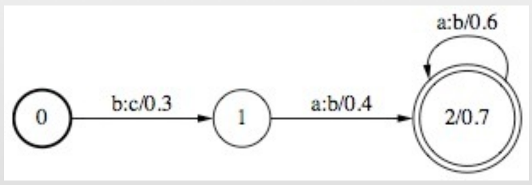
    $\delta_1(x,i:c) = v$ and $\delta_2(y,c:o) = z$

COMPOSITION OF TRANSDUCERS

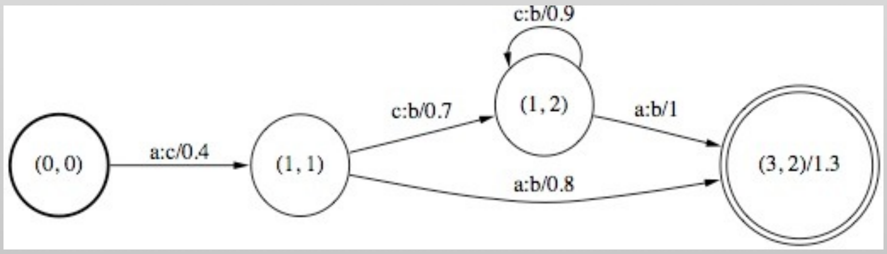
```
fstcompose A.fsm B.fsm > C.fsm
```



A.fst



B.fst



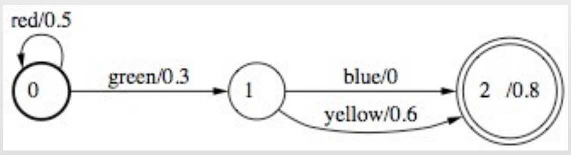
C.fst

INTERSECTION OF TRANSDUCERS

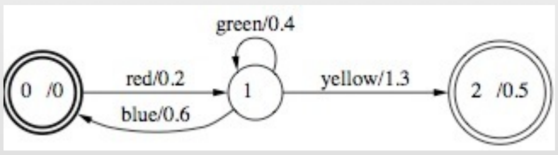
- *The intersection algorithm only considers the cartesian product of the states*
  - For each state q_i of the first transducer, and state q_j of the second transducer, build a new state q_{ij}
  - For the input symbol a, if the first transducer has a transition to the state q_n and the second transducer has a transition to state q_m the new transducer has a transition to state q_{nm}

INTERSECTION OF TRANSDUCERS

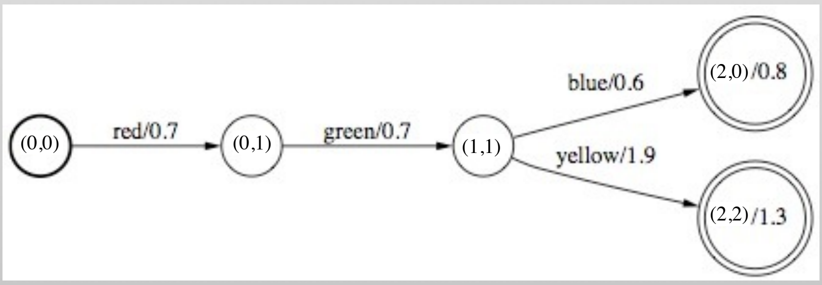
```
fstintersect A.fst B.fst > C.fst
```



A.fst



B.fst



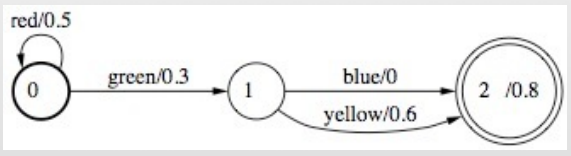
C.fst

DIFFERENCE OF TRANSDUCERS

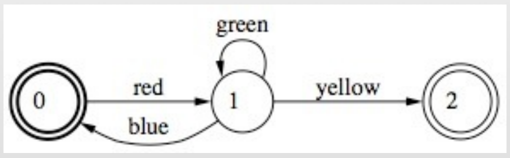
- *Difference(A,B) = Intersection(A,Complement(B))*
- *Complement(B) = all the sentences not belonging to B*

DIFFERENCE OF TRANSDUCERS

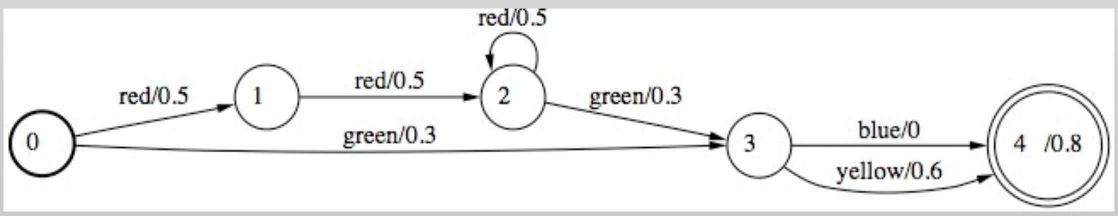
```
fsmdifference A.fsm B.fsm > C.fsm
```



A.fsm



B.fsm

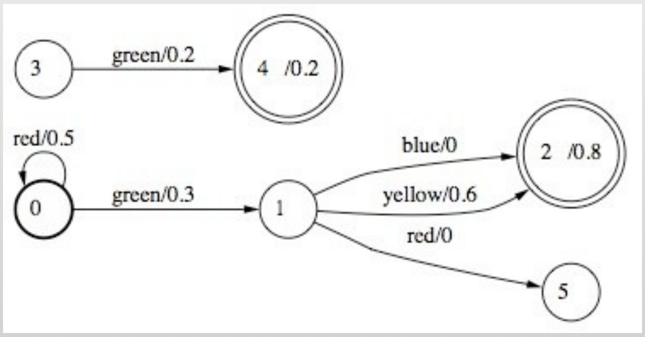


C.fsm

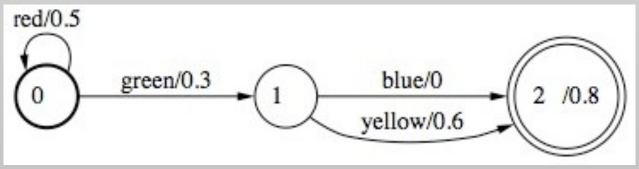
REMOVAL OF INACCESSIBLE STATES

- com a opção -t, devolve (exit status) 1 se a saída não tiver estados, útil para testar se a saída é vazia ...

```
fstconnect A.fst > C.fst
```



A.fst



C.fst