# WFSTs for ASR

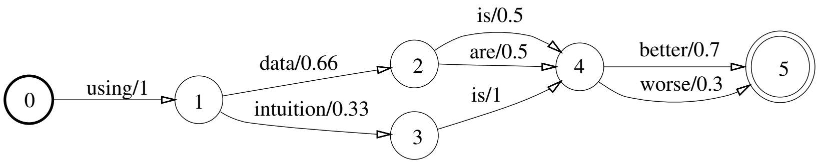
Peter Bell

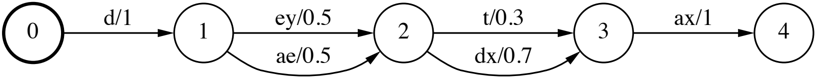
Automatic Speech Recognition – ASR Lecture 9
14 February 2022

# Weighted Finite State Transducers

- Weighted finite state automaton that transduces an input sequence to an output sequence (Mohri et al 2008)
- States connected by transitions. Each transition has
  - input label
  - output label
  - weight
- Weights use the *log semi-ring* or *tropical semi-ring* – with operations that correspond to multiplication and addition of probabilities
- There is a single start state. Any state can optionally be a final state (with a weight)
- Used by Kaldi

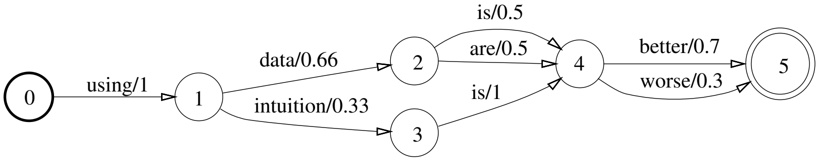
# Weighted Finite State Acceptors



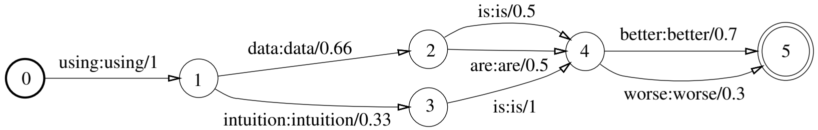



# Weighted Finite State Transducers

Acceptor

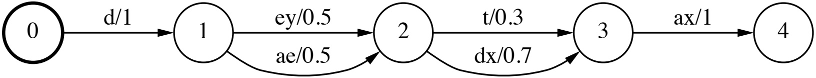



Transducer

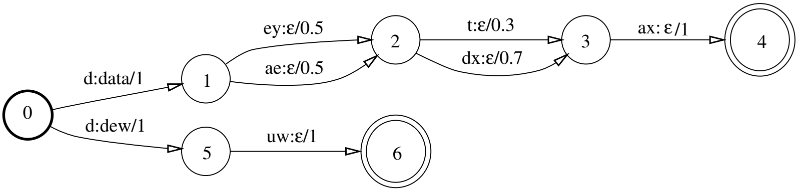


# Weighted Finite State Transducers

Acceptor




Transducer



# WFST Algorithms

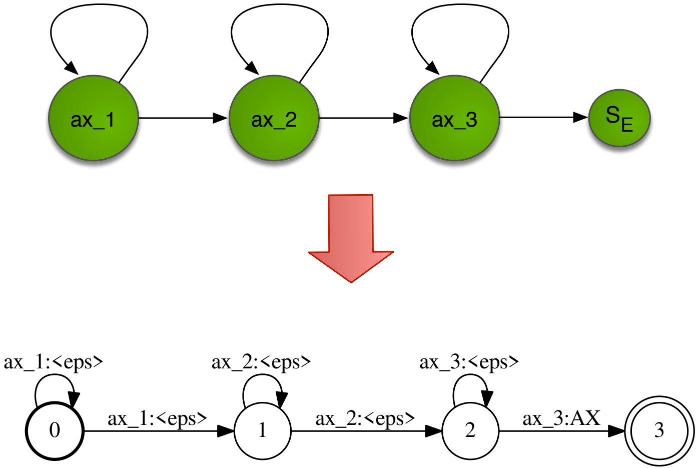
**Composition** Combine transducers $T_1$ and $T_2$ into a single transducer acting as if the output of $T_1$ was passed into $T_2$.

**Determinisation** Ensure that each state has no more than a single output transition for a given input label

**Minimisation** Transforms a transducer to an equivalent transducer with the fewest possible states and transitions

**Weight pushing** Push the weights towards the front of the path

# The HMM as a WFST



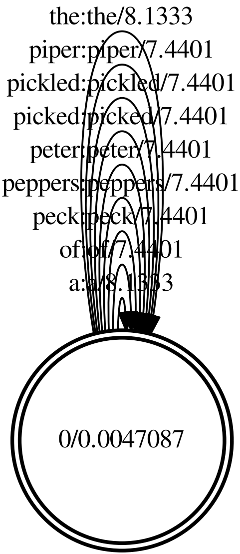
# Applying WFSTs to speech recognition

- Represent the following components as WFSTs

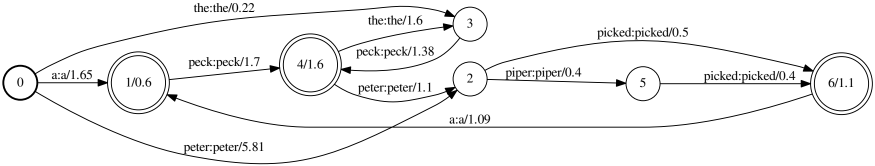
| | transducer | input sequence | output sequence |
|---|---|---|---|
| $G$ | word-level grammar | words | words |
| $L$ | pronunciation lexicon | phones | words |
| $C$ | context-dependency | CD phones | phones |
| $H$ | HMM | HMM states | CD phones |

- Composing $L$ and $G$ results in a transducer $L \circ G$ that maps a phone sequence to a word sequence
- $H \circ C \circ L \circ G$ results in a transducer that maps from HMM states to a word sequence

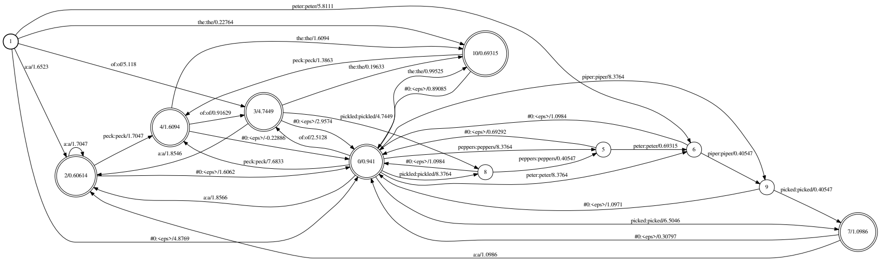
# Grammar - unigram



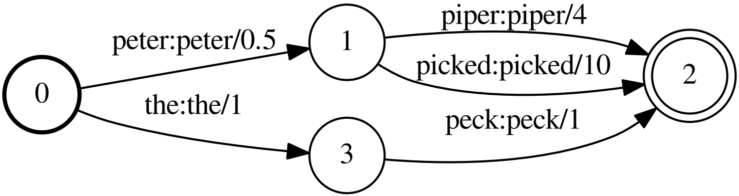
# Grammar - bigram



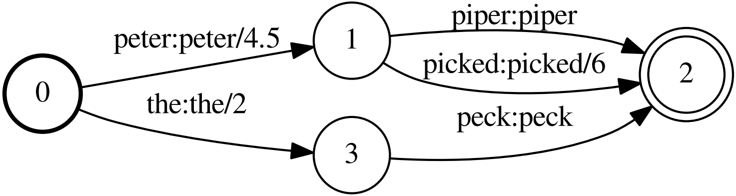
# Bigram with back-off



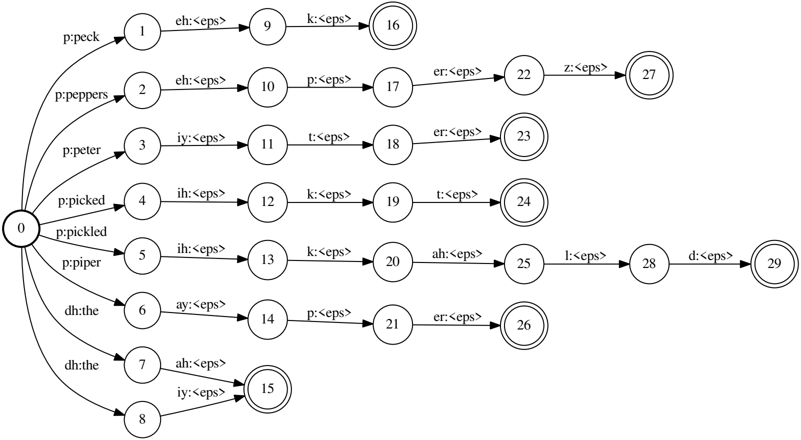
# A toy example



# Weight-pushed version

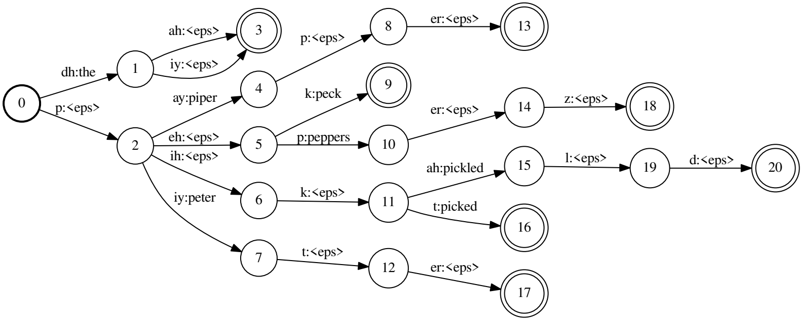


# Lexicon, *L*

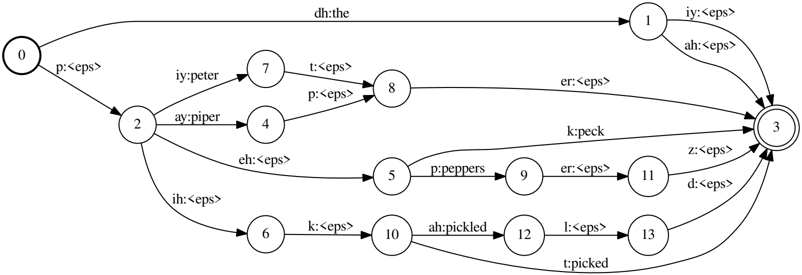


0
p:peck
1
eh:<eps>
9
k:<eps>
16
p:peppers
2
eh:<eps>
10
p:<eps>
17
er:<eps>
22
z:<eps>
27
p:peter
3
iy:<eps>
11
t:<eps>
18
er:<eps>
23
p:picked
4
ih:<eps>
12
k:<eps>
19
t:<eps>
24
p:pickled
5
ih:<eps>
13
k:<eps>
20
ah:<eps>
25
l:<eps>
28
d:<eps>
29
p:piper
6
ay:<eps>
14
p:<eps>
21
er:<eps>
26
dh:the
7
ah:<eps>
15
dh:the
8
iy:<eps>

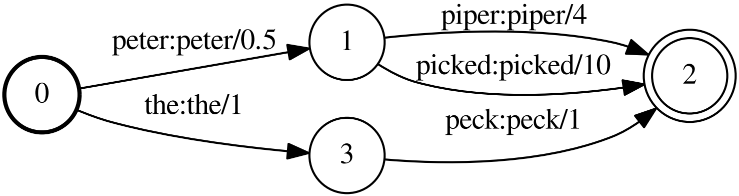
# Determinization – $det(L)$



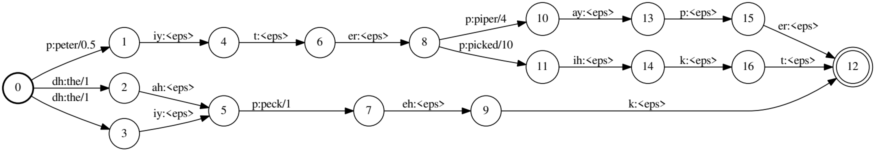
# Minimization – *min*(*det*(*L*))



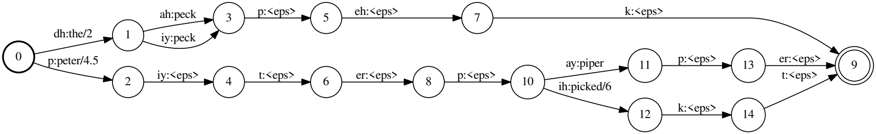
# Composition



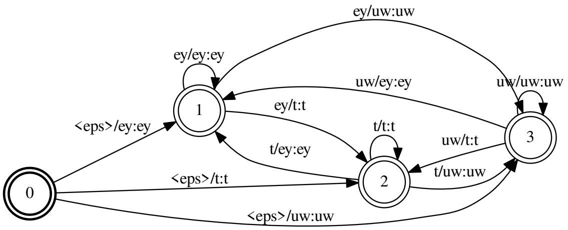
# Composition: $L \circ G$



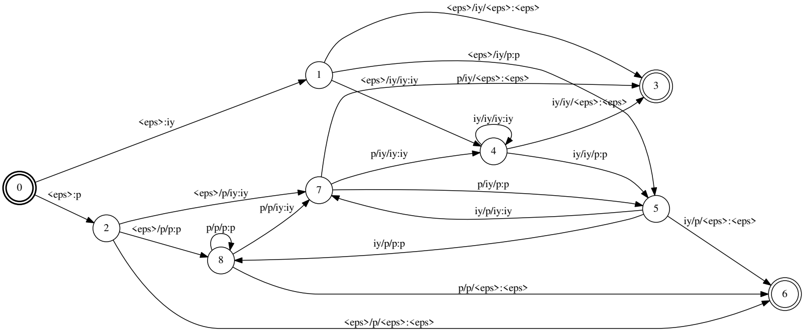
# $min(det(L \circ G))$



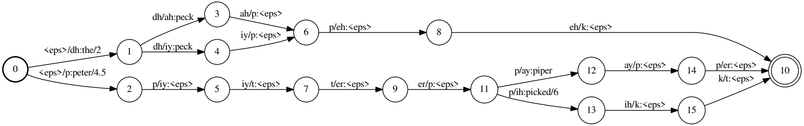
# Context-dependency: left biphones



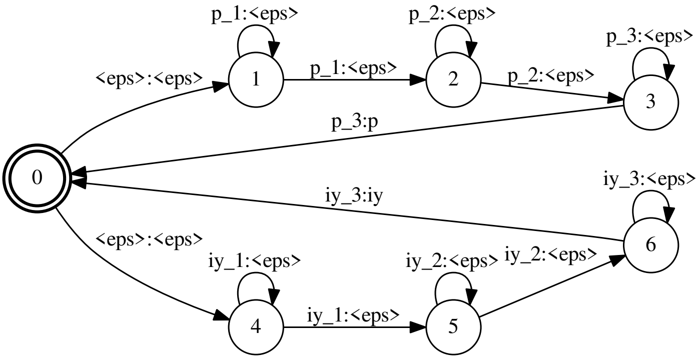
# Context-dependency: triphones



# $C \circ L \circ G$ – biphones



# HMM transducer, *H*




- We can also use a version that outputs context-dependent phones
- *H* can be used to encode state-tying

# Decoding using WFSTs

- Combining the transducers gives an overall HMM structure for the ASR system – but minimisation and determination operations on the WFSTs means it is much smaller than naively combining the HMMs
- But it is important in which order the algorithms are combined otherwise the transducers may "blow-up"
- standard approach is to determinize and minimize after each composition
- In Kaldi, ignoring one or two details

$$HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G))))))$$

# Reading

- Mohri et al (2008). "Speech recognition with weighted finite-state transducers." In Springer Handbook of Speech Processing, pp. 559-584. Springer.
  http://www.cs.nyu.edu/~mohri/pub/hbka.pdf
- WFSTs in Kaldi. http://danielpovey.com/files/Lecture4.pdf