Finite State Transducers. . .
for Dummies

Stanley Chen

Carnegie
Mellon

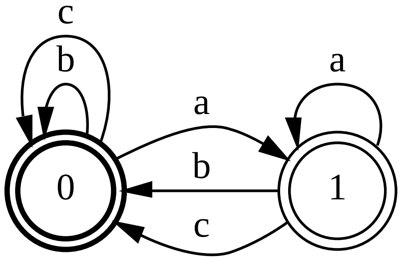
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

April 9, 1998

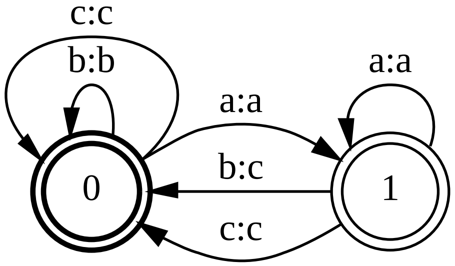
Introduction

What is a finite-state transducer (FST)?

- it's a finite-state automaton...



- with output labels

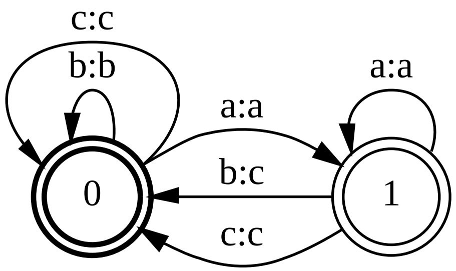


Introduction

What do finite-state transducers do?

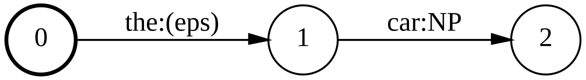
- rewrite strings

abcb ⇒

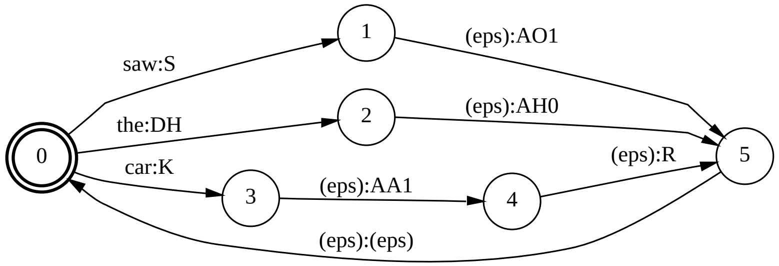


⇒ accb

saw the car ⇒ · · ·



· · · ⇒ saw NP



saw the car ⇒ ⇒ S AO1 DH AH0 K AA1 R

Introduction

Why should you, as an ace speech recognition researcher, care about finite-state transducers?

- the process of *decoding* can be formulated elegantly as the application of a series of FST's

$$\text{acoustic signal} \overset{\substack{\text{phone}\\ \text{models}}}{\Rightarrow} \text{phone lattice} \overset{\text{dict}}{\Rightarrow} \text{word lattice} \overset{\text{LM}}{\Rightarrow} \text{word lattice}$$

- AT&T Research has released a set of efficient and powerful tools for manipulating FSA's and FST's
- $\Rightarrow$ it becomes possible to apply different types of models without constructing new decoders (*e.g.*, quinphone, 4-gram)

Overview

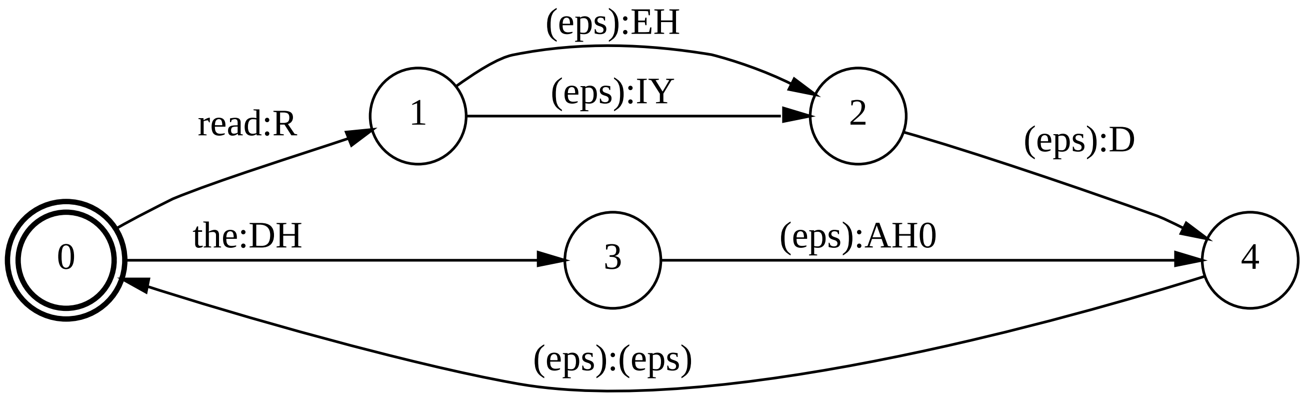
- more about FST's
  - string rewriting $\Rightarrow$ lattice rewriting
  - weighted FST's
- decoding as finite-state transduction
- FSM, the AT&T finite-state machine library
  - efficient?
- discussion
  - other applications of FST's

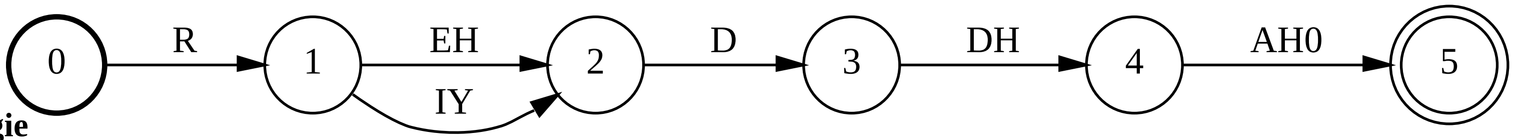
Finite-State Transducers

nondeterminism

- a single string may be rewritten in multiple ways
- can express set of output strings as FSA

read the





Finite-State Transducers

application to sets of strings (lattices)

- given set of strings expressed as FSA. . .
- efficient algorithm for finding all possible output strings, expressed as FSA

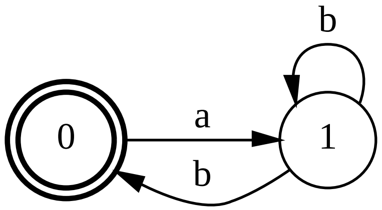
Finite-State Transducers

application/composition

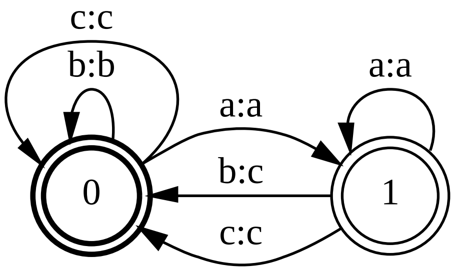
- given FSA with states $A = \{a_1, \ldots, a_n\}$
- apply FST with states $T = \{t_1, \ldots, t_m\}$
- resulting FSA has states $A \times T$
- edge from (a_{i_1}, t_{j_1}) to (a_{i_2}, t_{j_2}) with label l_{out} iff there is a label l_{in} such that
  - edge from a_{i_1} to a_{i_2} with label l_{in}
  - edge from t_{j_1} to t_{j_2} with label $l_{\text{in}} : l_{\text{out}}$

Finite-State Transducers

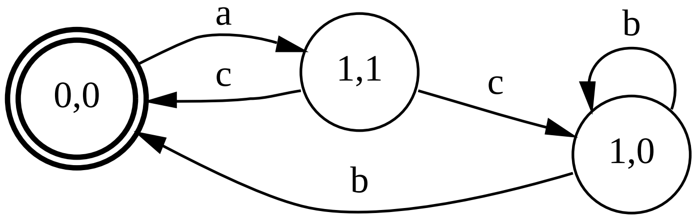
- $(ab^{*}b)^{*}$



- change b to c if after an a



- $(ac|acb^{*}b)^{*}$



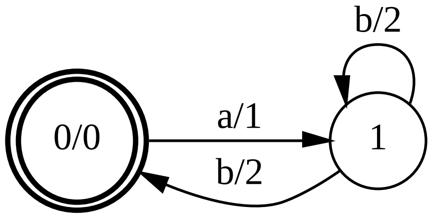
Finite-State Transducers

weighted FSA's/FST's (WFSA/WFST) (a.k.a. lattices)

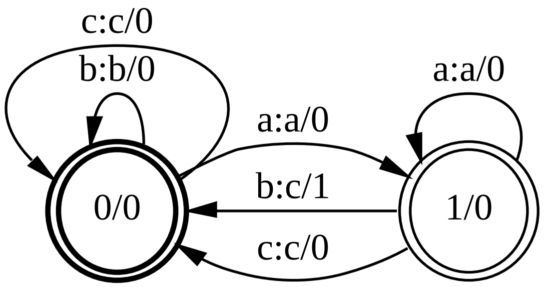
- weight/score of edge is a log prob
- weight/score of path is sum of weights of edges
- find path with max weight/score
- when apply FST, add weights
- other interpretations possible (any semiring)

Finite-State Transducers

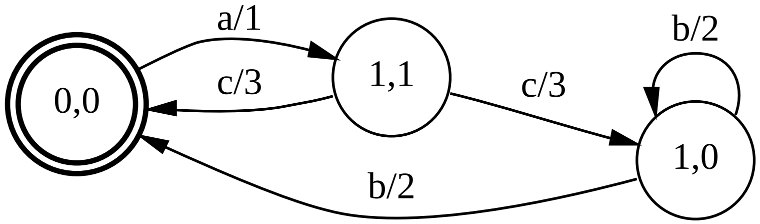
- $(ab^*b)^*$



- change b to c if after an a



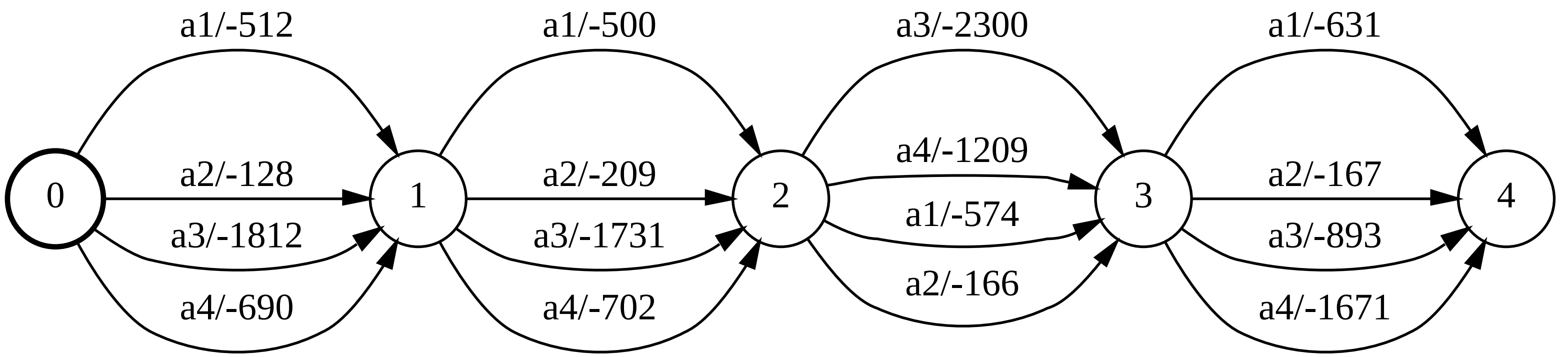
- $(ac|acb^*b)^*$



Decoding as Transduction

input: senone lattice

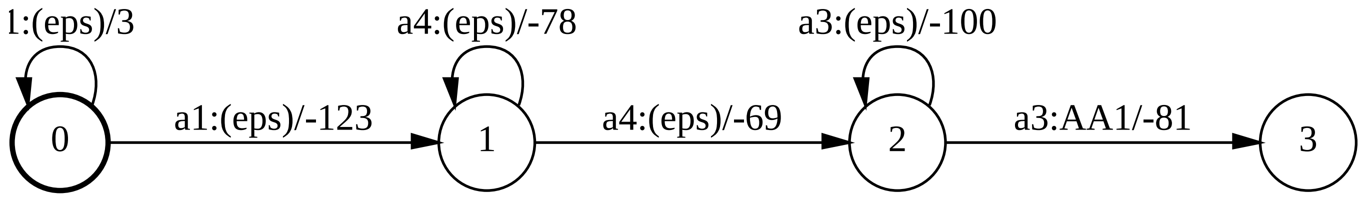
- say 4 senones: $a_1, \ldots, a_4$
- $s_{i,t}$ is score/logprob of senone a_i generating frame t



Decoding as Transduction

transduction: senone lattice ⇒ phone lattice

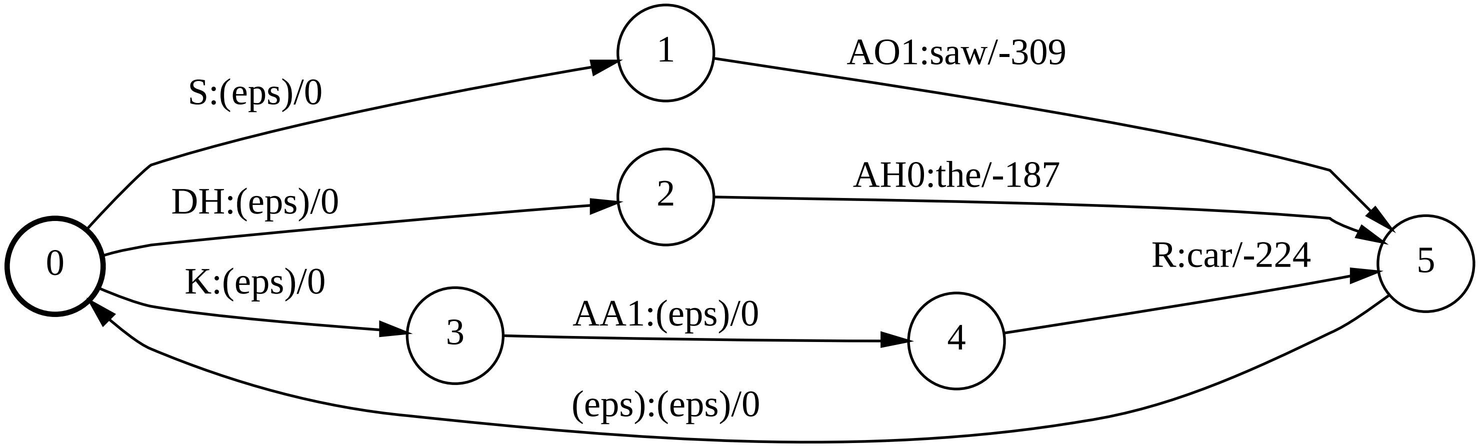
- phone models
- edges enforce context-dependence, *e.g.*, triphones



Decoding as Transduction

transduction: phone lattice ⇒ word lattice

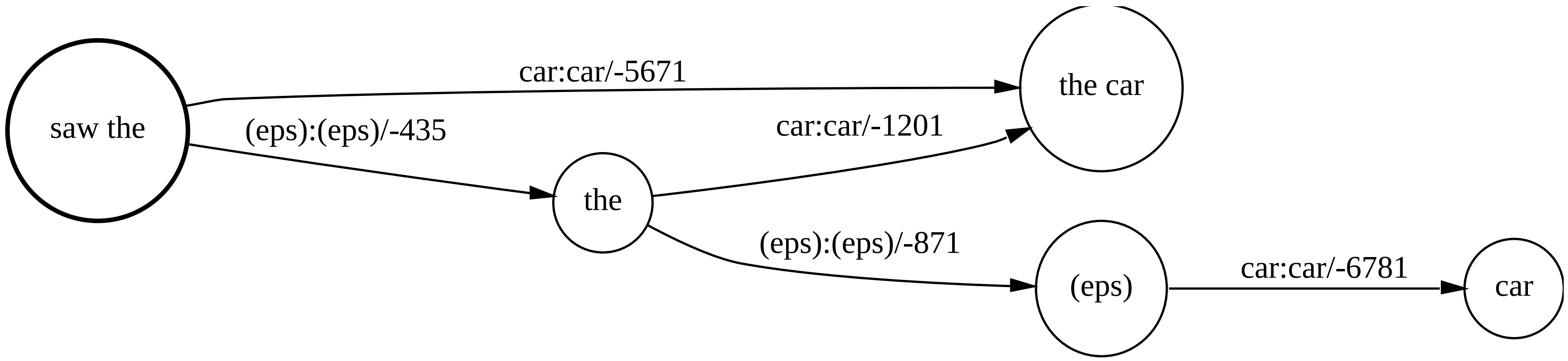
- dictionary



Decoding as Transduction

transduction: word lattice ⇒ word lattice

- language model



Decoding as Transduction

theory

- L = LM, D = dict, A = phone models, O = observations/senone lattice

$$\text{hyp} = \text{best-path}(L(D(A(O))))$$

practice

- composition

$$\text{hyp} = \text{best-path}((L \circ D \circ A)(O))$$

- pruning

$$\text{hyp} = \text{best-path}((L \circ D)(\text{prune}(A(\text{prune}(O)))))$$

FSA/FST Tools

efficient algorithms for WFSA/WFST (Mohri, Pereira, Riley)

- determinization (if possible)
- minimization

FSM

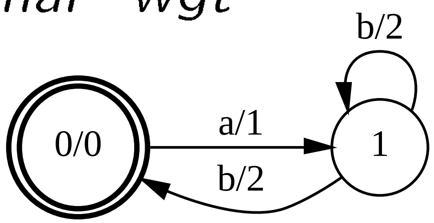
- publicly availaible toolkit from AT&T by above
- `http://www.research.att.com/sw/tools/`
- *graphviz*, tools for visualizing graphs

FSA/FST Tools

FSM input

- start with WFSA/WFST in simple text format

src	*dest*	*label*	*wgt*
0	1	a	1
1	0	b	2
1	1	b	2
0	0		
final	*wgt*		



FSA/FST Tools

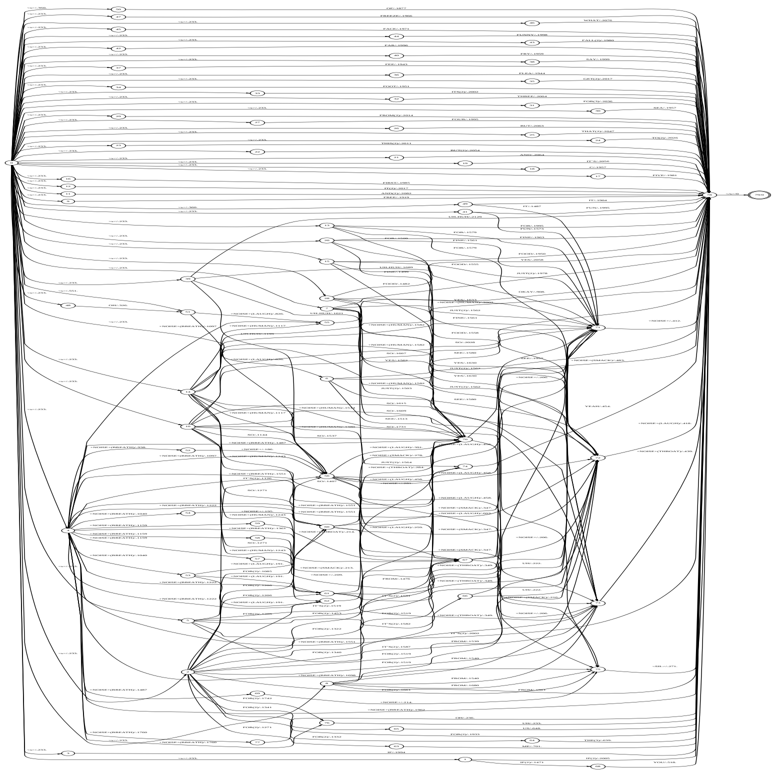
FSM operations: given WFSA's A_1, A_2; WFST's T_1, T_2

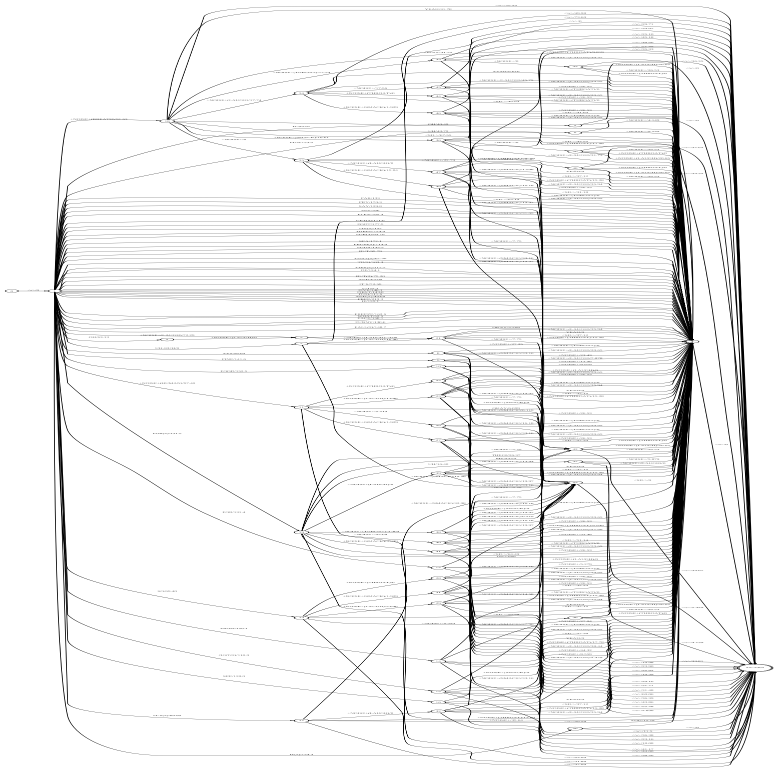
- application $T_1(A_1)$, composition $T_1 \circ T_2$
- determinization, minimization
- best path, n-best lists, pruning (`s3dag`, `s3astar`)
- generation
- union $A_1 \cup A_2$, intersection $A_1 \cap A_2$, concatenation $A_1 A_2$
- and much, much more!

FSA/FST Tools

real-life example: determinization/minimization of Janus lattice

- before/after





FSA/FST Tools

how efficient is FSM?

- big transducers, slow transduction? lazy expansion
- Johns Hopkins
- 1995 Hub 3 (NAB), off-line system
  - 60k vocab, 5-gram language model
  - 90.5% WA, 500$\times$ RT
  - multi-pass
- 1995 Hub 3 (NAB), real-time system
  - 5k senones, 4 mixtures/senone, trigram LM with 0-2-4+ cutoffs
  - 81.2% WA, 2.2$\times$ RT
  - single-pass

Discussion

other applications

- speech synthesis
- parsing (context-sensitive)
- part-of-speech tagging
- word alignment/oracle WER calculation
- anything that can be interpreted as transduction/string rewriting

Discussion

FST's

- can vary structure of models without writing new decoder, if can express as FST (and can train it)
  - 3-state v. 5-state HMM's, quinphones, pronunication networks, 4-gram
- AT&T tools are a good thing
- unclear whether AT&T tools are efficient enough for prime time