

Finite-State Techniques for Speech Recognition

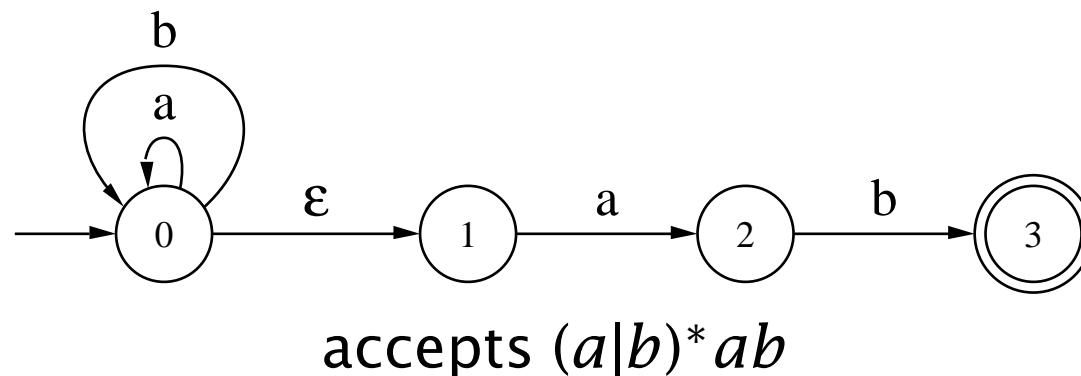
- motivation
- definitions
 - finite-state acceptor (FSA)
 - finite-state transducer (FST)
 - deterministic FSA/FST
 - weighted FSA/FST
- operations
 - closure, union, concatenation
 - intersection, composition
 - epsilon removal, determinization, minimization
- on-the-fly implementation
- FSTs in speech recognition: recognition cascade
- research systems within SLS impacted by FST framework
- conclusion

MIT

Motivation

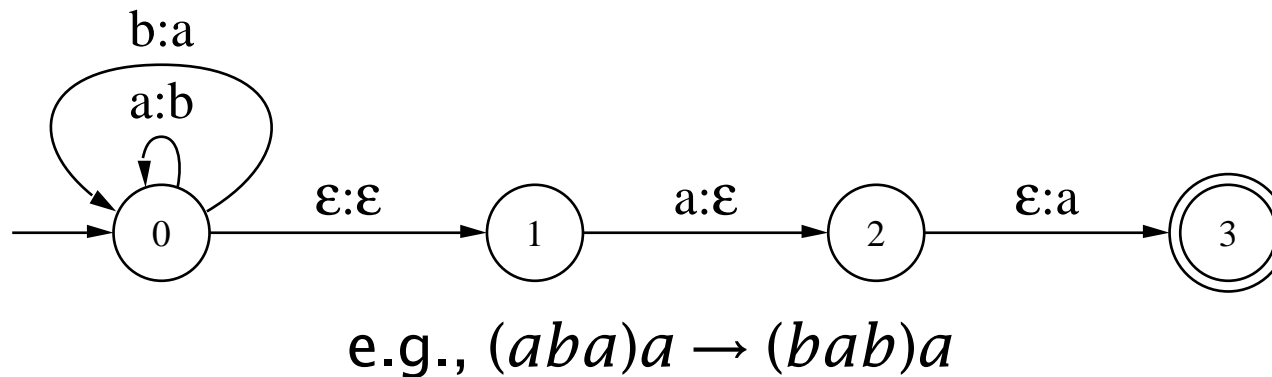
- many speech recognition components/constraints are finite-state
 - language models (e.g., n -grams, on-the-fly CFGs)
 - lexicons
 - phonological rules
 - N -best lists
 - word graphs
 - recognition paths
- should use same representation and algorithms for all
 - consistency
 - make powerful algorithms available at all levels
 - flexibility to combine or factor in unforeseen ways
- AT&T [Pereira, Riley, Ljolje, Mohri, et al.]

Finite-State Acceptor (FSA)



- definition:
 - finite number of states
 - one initial state
 - at least one final state
 - transition labels:
 - * label from alphabet Σ must match input symbol
 - * ϵ consumes no input
- accepts a regular language

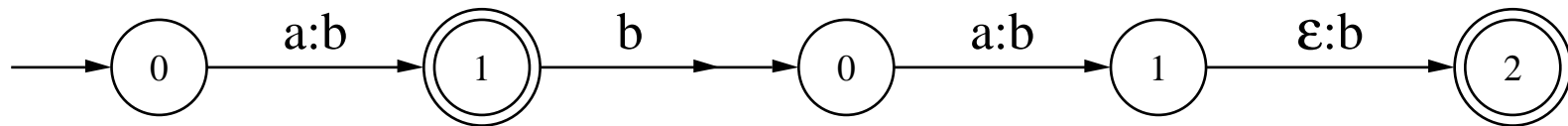
Finite-State Transducer (FST)



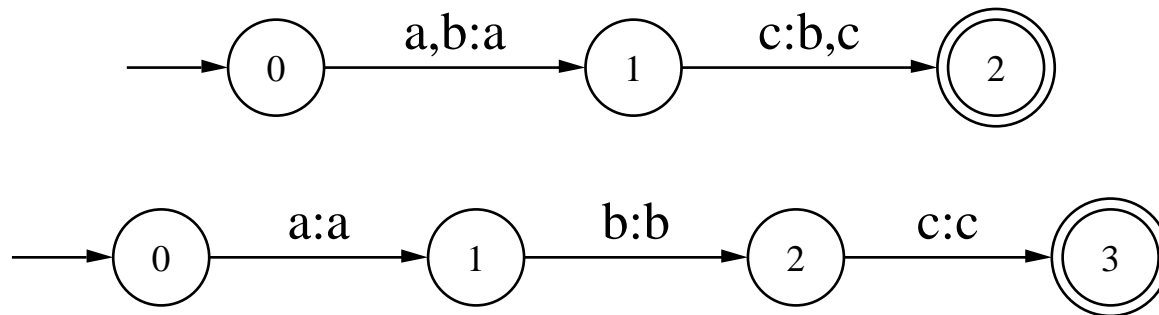
- definition, like FSA except:
 - transition labels:
 - * *pairs* of input:output labels
 - * ϵ on input consumes no input
 - * ϵ on output produces no output
- relates input sequences to output sequences (maybe ambiguous)
- FST with labels $x:x$ is an FSA

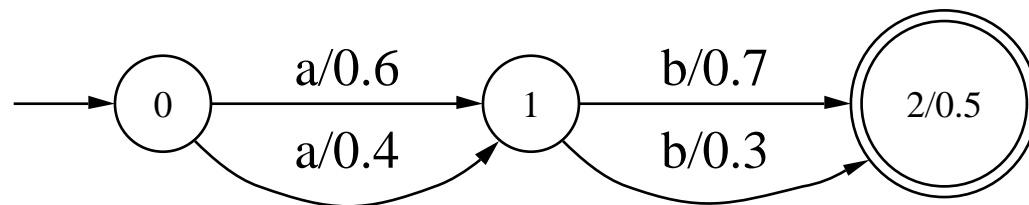
Finite-State Transducer (FST)

- final states can have outputs, but we use ϵ transitions instead

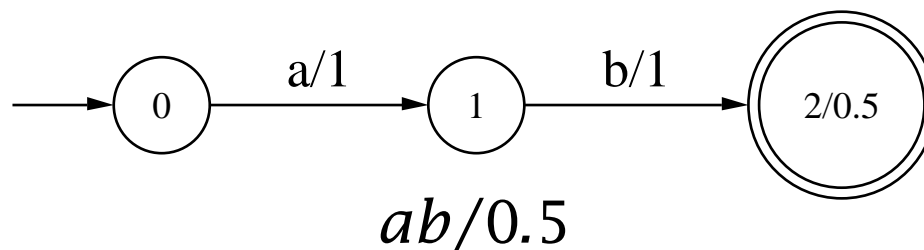


- transitions can have multiple labels, but we split them up

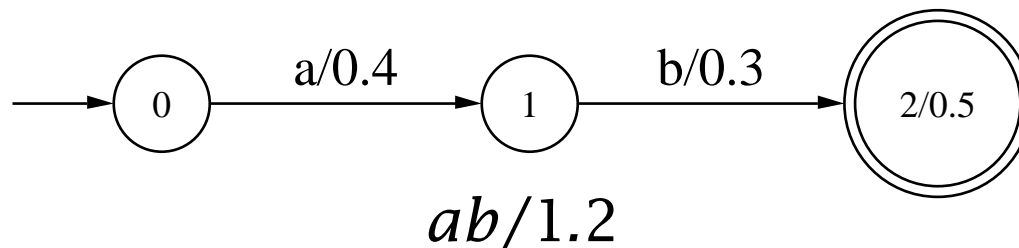




- transitions and final states can have weights (costs or scores)
- weight *semirings* $(\oplus, \otimes, \mathbf{0}, \mathbf{1})$, $\oplus \sim$ parallel, $\otimes \sim$ series:
 - $\mathbf{0} \oplus x = x$, $\mathbf{1} \otimes x = x$, $\mathbf{0} \otimes x = \mathbf{0}$, $\mathbf{0} \otimes \mathbf{1} = \mathbf{0}$
 - $(+, \times, 0, 1) \sim$ probability (sum parallel, multiply series)

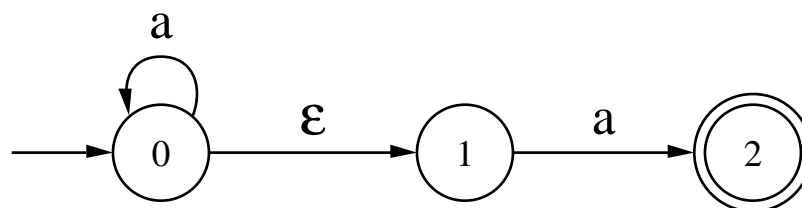


- $(\min, +, \infty, 0) \sim -\log$ probability (best of parallel, sum series)

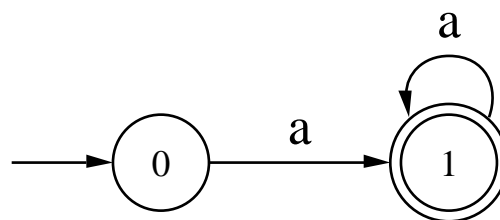


Deterministic FSA or FST

- input sequence uniquely determines state sequence
- no ϵ transitions
- at most one transition per label for all states



non-deterministic (NFA)



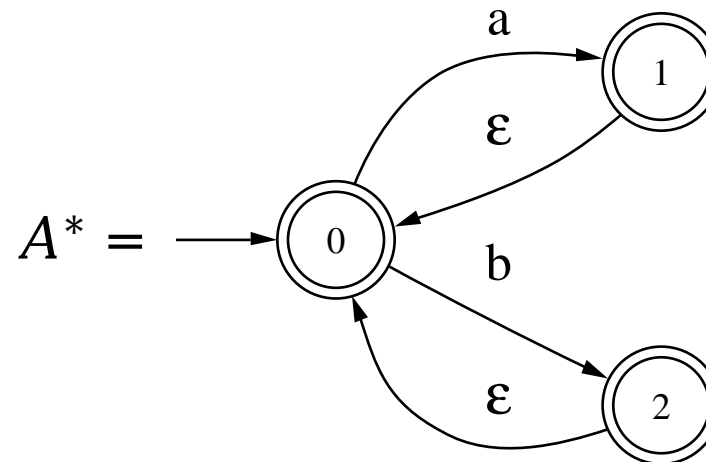
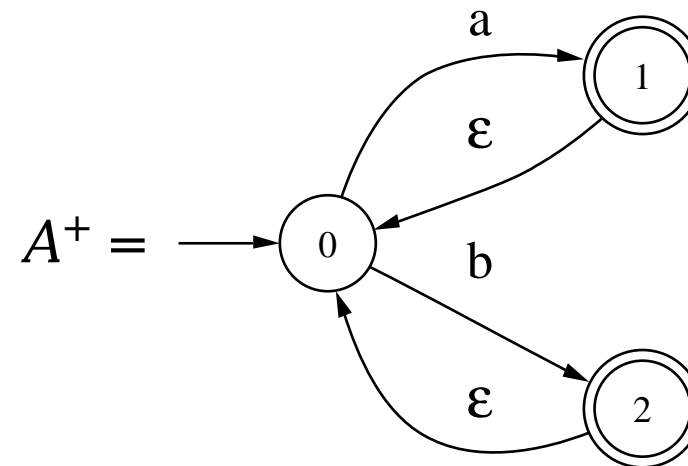
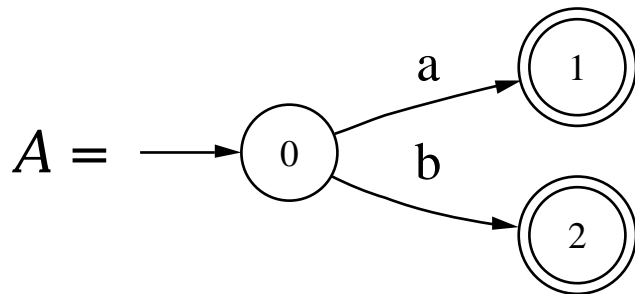
deterministic (DFA)

MIT

Operations

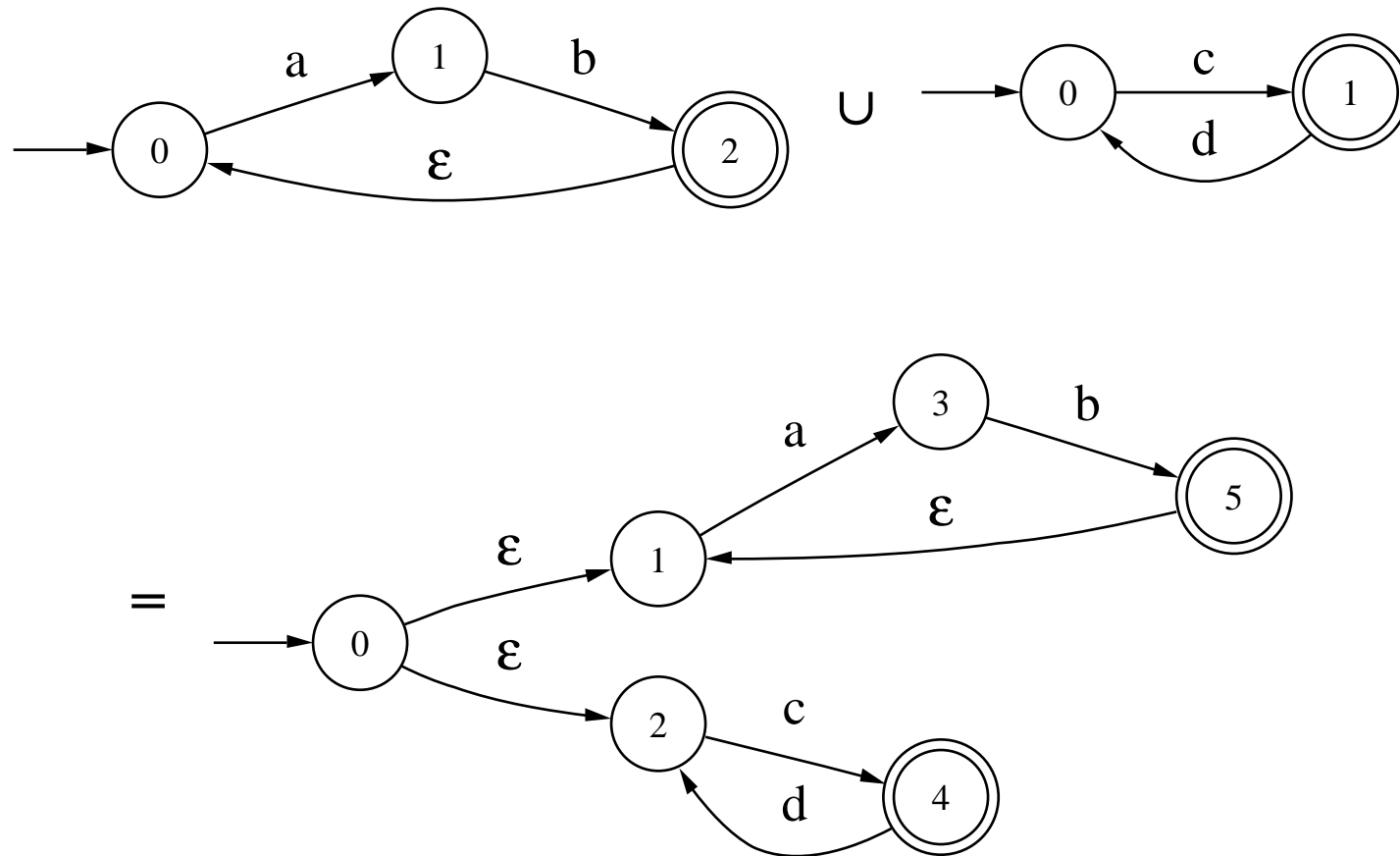
- constructive operations:
 - closure A^* and A^+
 - union $A \cup B$
 - concatenation AB
 - complementation \bar{A} (FSA only)
 - intersection $A \cap B$ (FSA only)
 - composition $A \circ B$ (FST only, FSA $\equiv \cap$)
- identity operations (optimization):
 - epsilon removal
 - determinization
 - minimization

Closure: A^+ , A^*



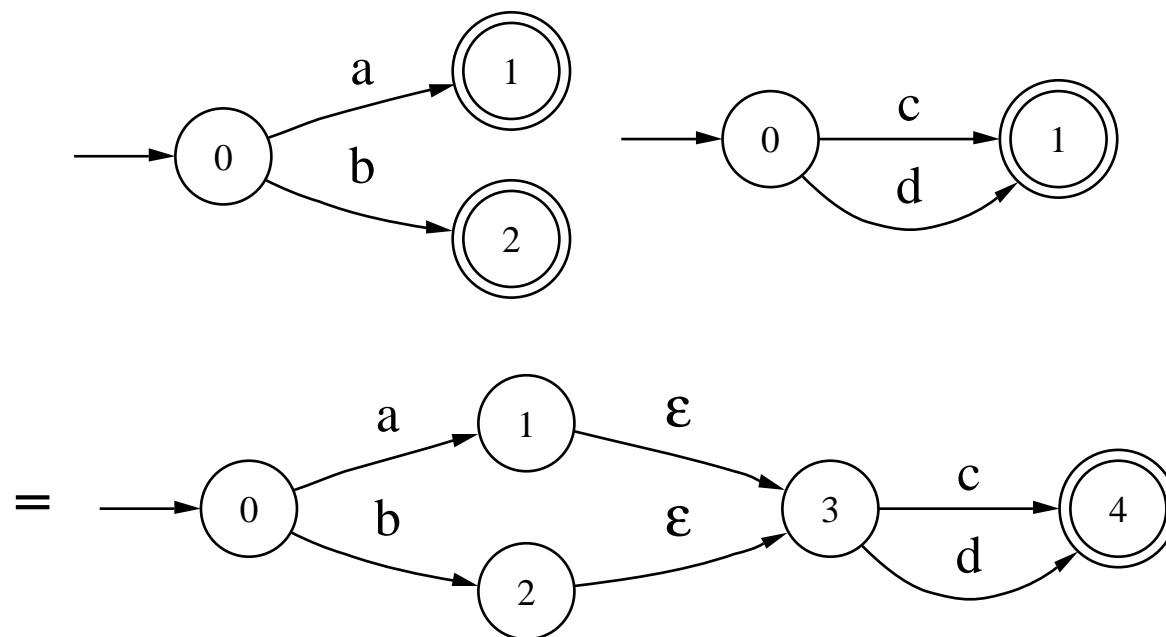
Union: $A \cup B$

parallel combination, e.g.,



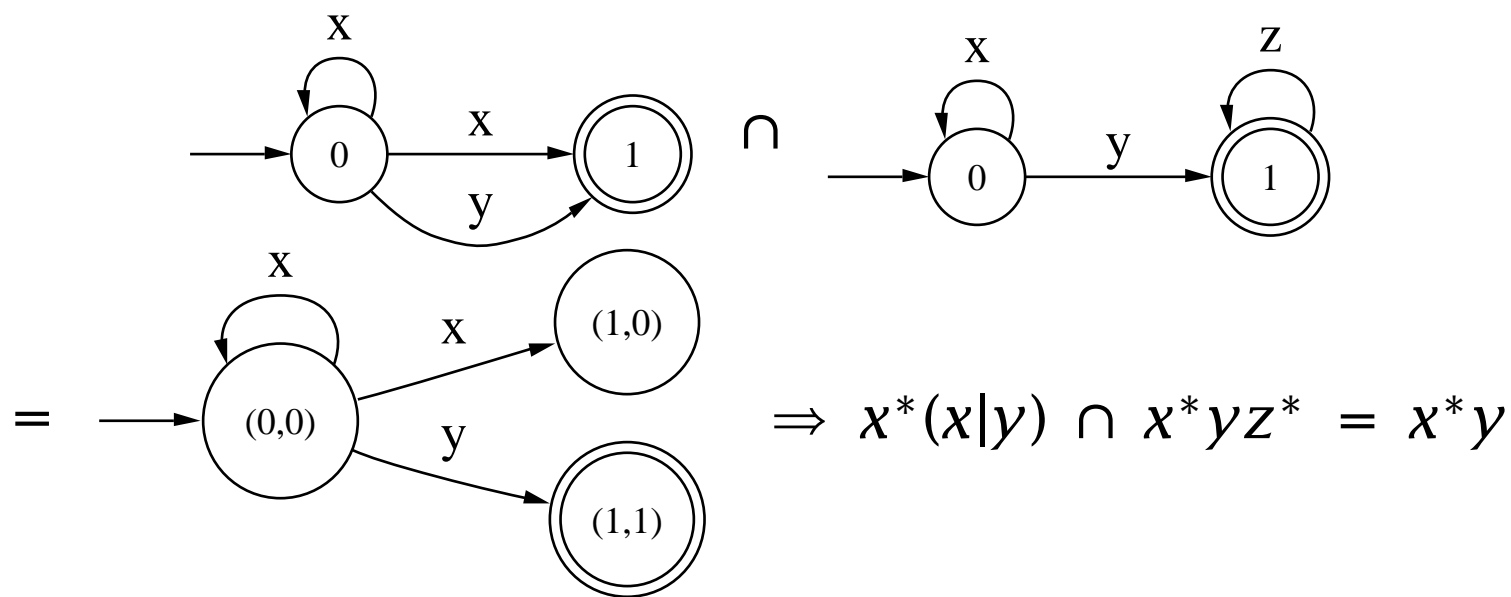
Concatenation: AB

serial combination, e.g.,



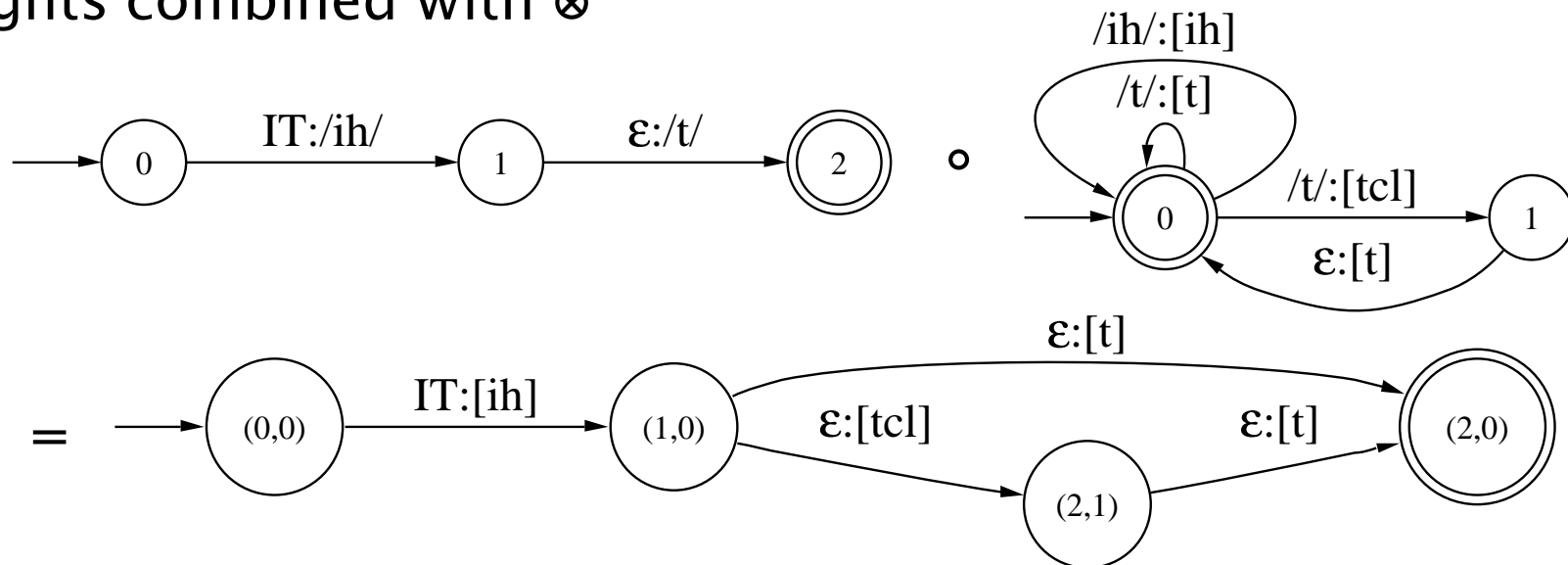
FSA Intersection: $A \cap B$

- output states associated with input state pairs (a, b)
- output state is final only if both a and b are final
- transition with label x only if both a and b have x transition
- weights combined with \otimes



FST Composition: $A \circ B$

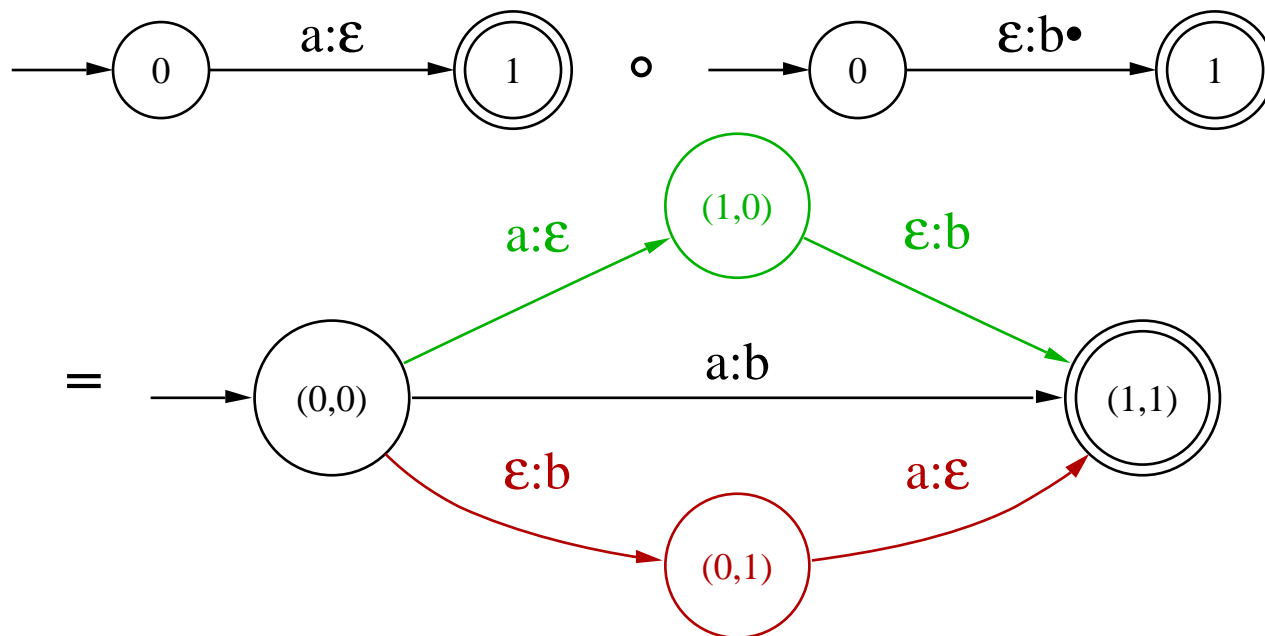
- output states associated with input state pairs (a, b)
- output state is final only if both a and b are final
- transition with label $x:y$ only if a has $x:\alpha$ and b has $\alpha:y$ transition
- weights combined with \otimes



- $(\text{words} \rightarrow \text{phonemes}) \circ (\text{phonemes} \rightarrow \text{phones}) = (\text{words} \rightarrow \text{phones})$

FST Composition: ϵ Interaction

- A output ϵ allows B to hold
- B input ϵ allows A to hold



- multiple paths typically filtered (resulting in dead end states)

FST Composition: Parsing

- language model from JSGF grammar compiled into on-the-fly recursive transition network (RTN) transducer G :

```

<top> = <forecast> | <conditions> | ... ;
<forecast> = [what is the] forecast for <city> {FORECAST};
<city> = boston [massachusetts] {BOS}
        | chicago [illinois] {ORD};

```

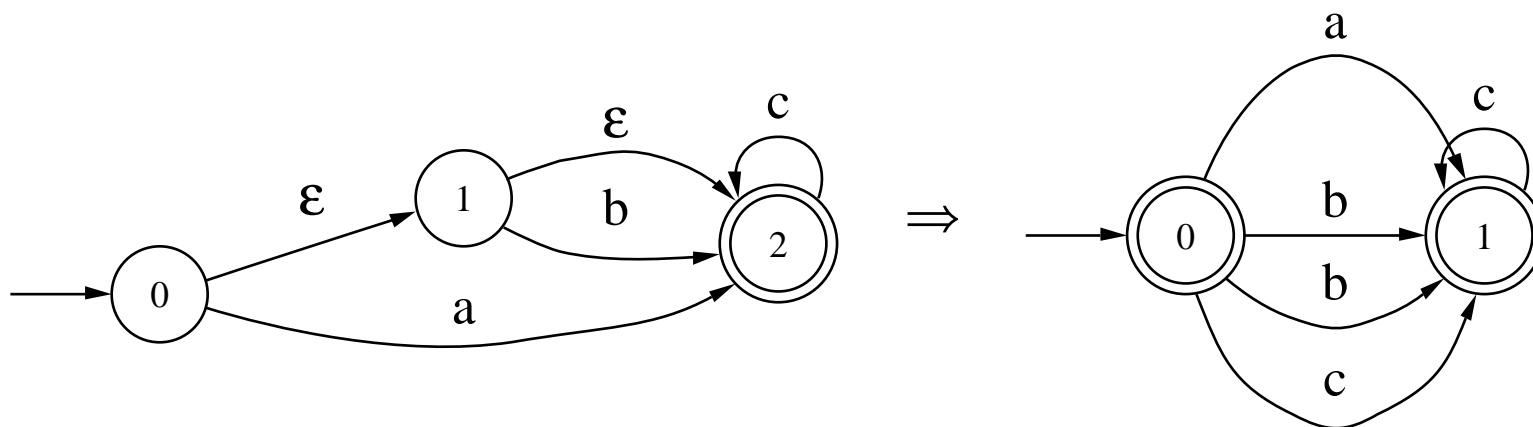
- “what is the forecast for boston” $\circ G \rightarrow$
 - BOS FORECAST *output tags only*
 - <forecast> what is the forecast for <city> boston </city>
</forecast> *bracketed parse*

FST Composition Summary

- very powerful operation
- can implement other operations:
 - intersection
 - application of rules/transformations
 - instantiation
 - dictionary lookup
 - parsing

Epsilon Removal (Identity)

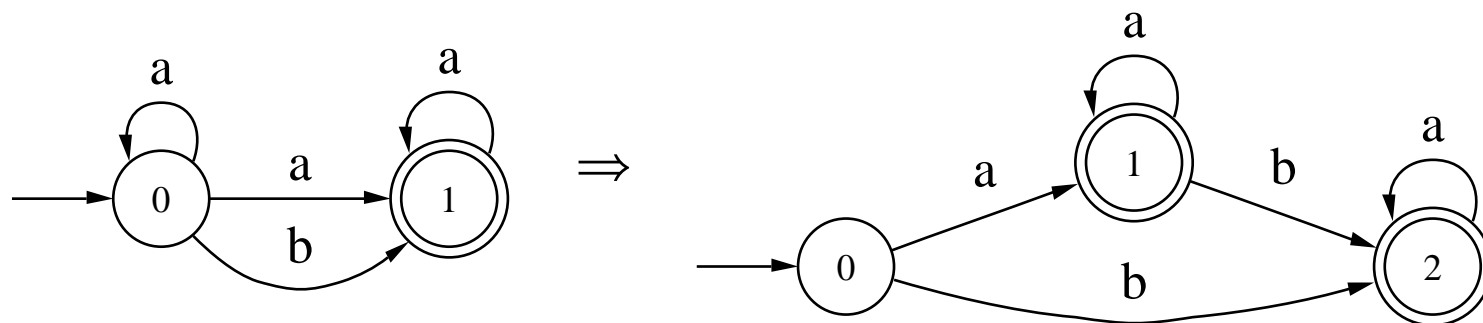
- required for determinization
- compute ϵ -closure for each state: set of states reachable via ϵ^*



- can dramatically increase number of transitions (copies)

FSA Determinization (Identity)

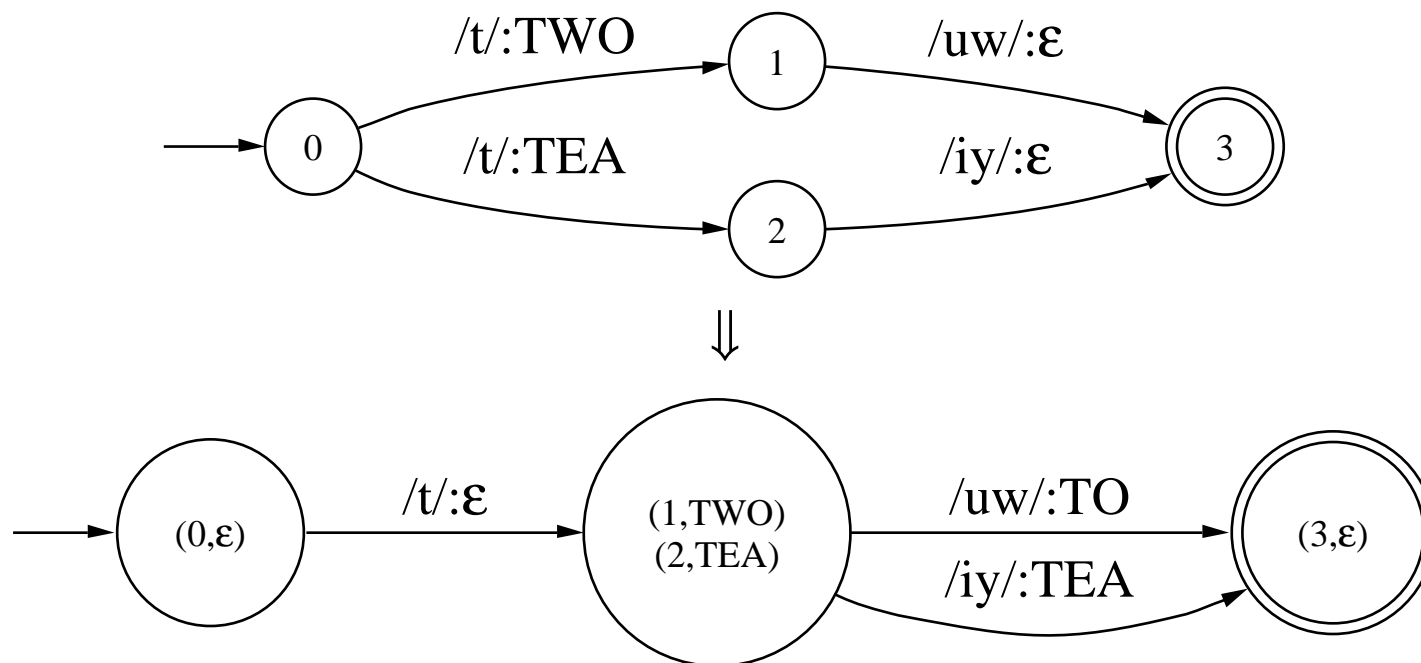
- subset construction
 - output states associated with *subsets* of input states
 - treat a subset as a superposition of its states
- worst case is exponential (2^N)
- locally optimal: each state has at most $|\Sigma|$ transitions



- weights: subsets of (state, weight)
 - weights might be delayed
 - transition weight is \oplus subset weights
 - worst case is infinite (not common)

FST Determinization (Identity)

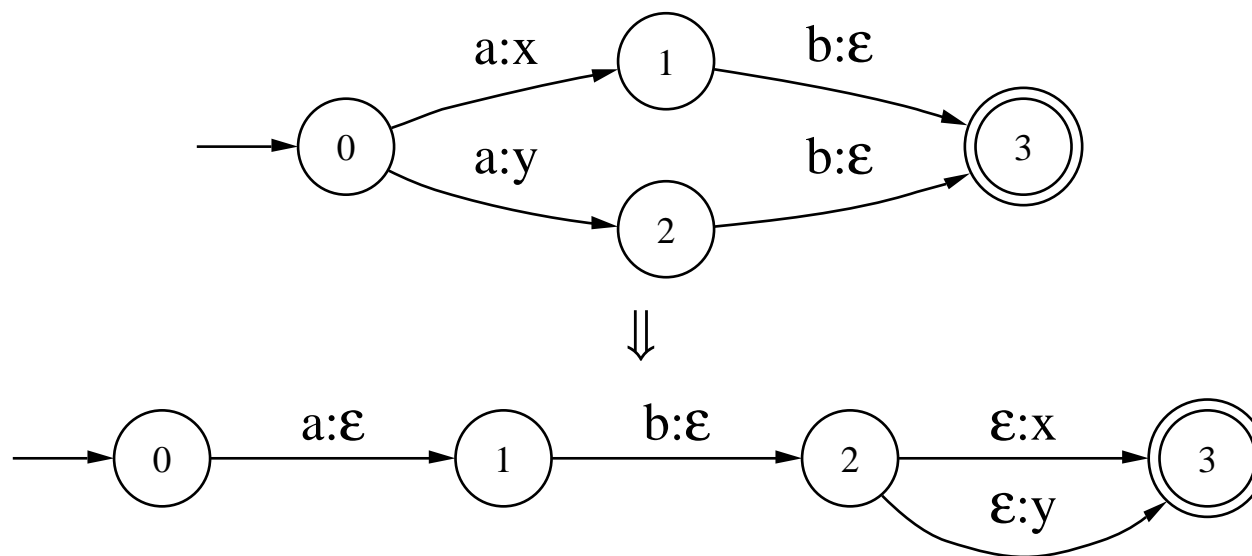
- subsets of (state, output*, weight)
- outputs and weights might be delayed
- transition output is least common prefix of subset outputs



- worst case is infinite (not uncommon due to ambiguity)

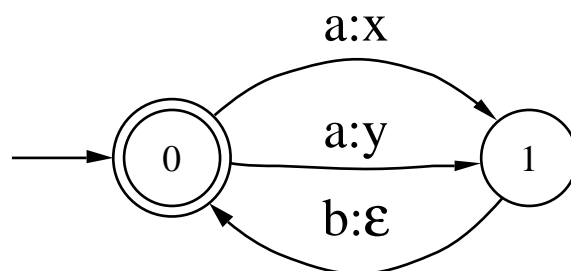
MIT FST Ambiguity

- input sequence maps to more than one output (e.g., homophones)
- finite ambiguity (delayed to output states):

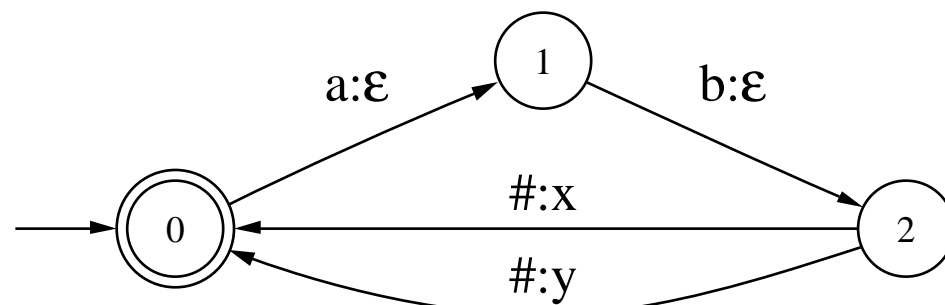
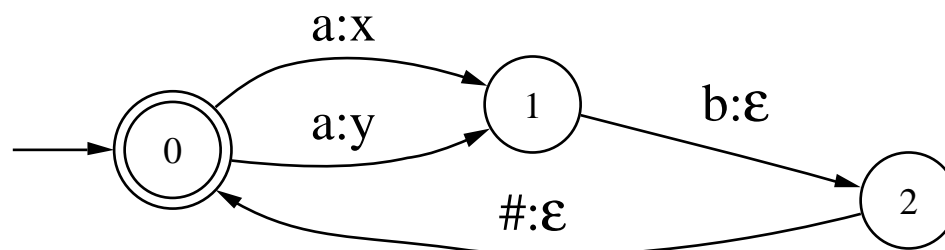


FST Ambiguity

- cycles (e.g., closure) can produce infinite ambiguity
- infinite ambiguity (cannot be determinized):

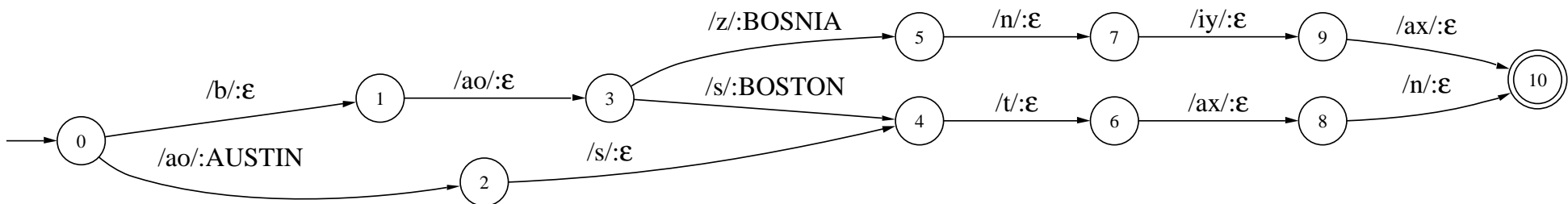
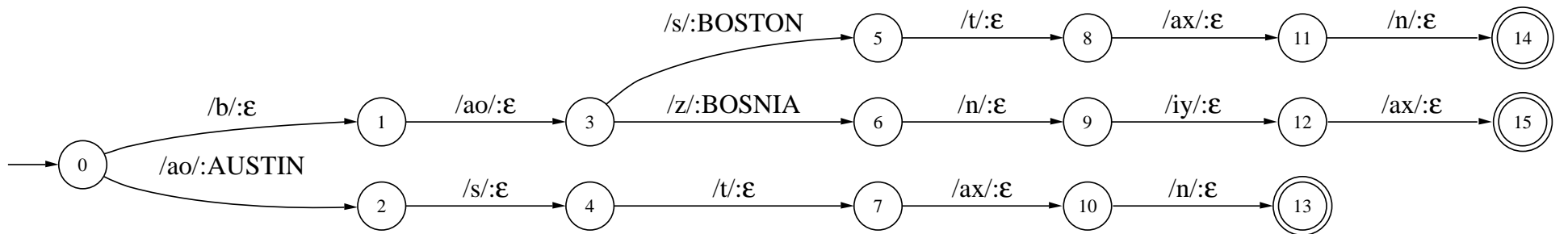


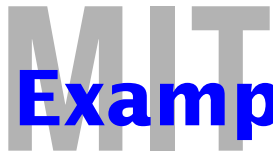
- a solution: our implementation forces outputs at #, a special ϵ



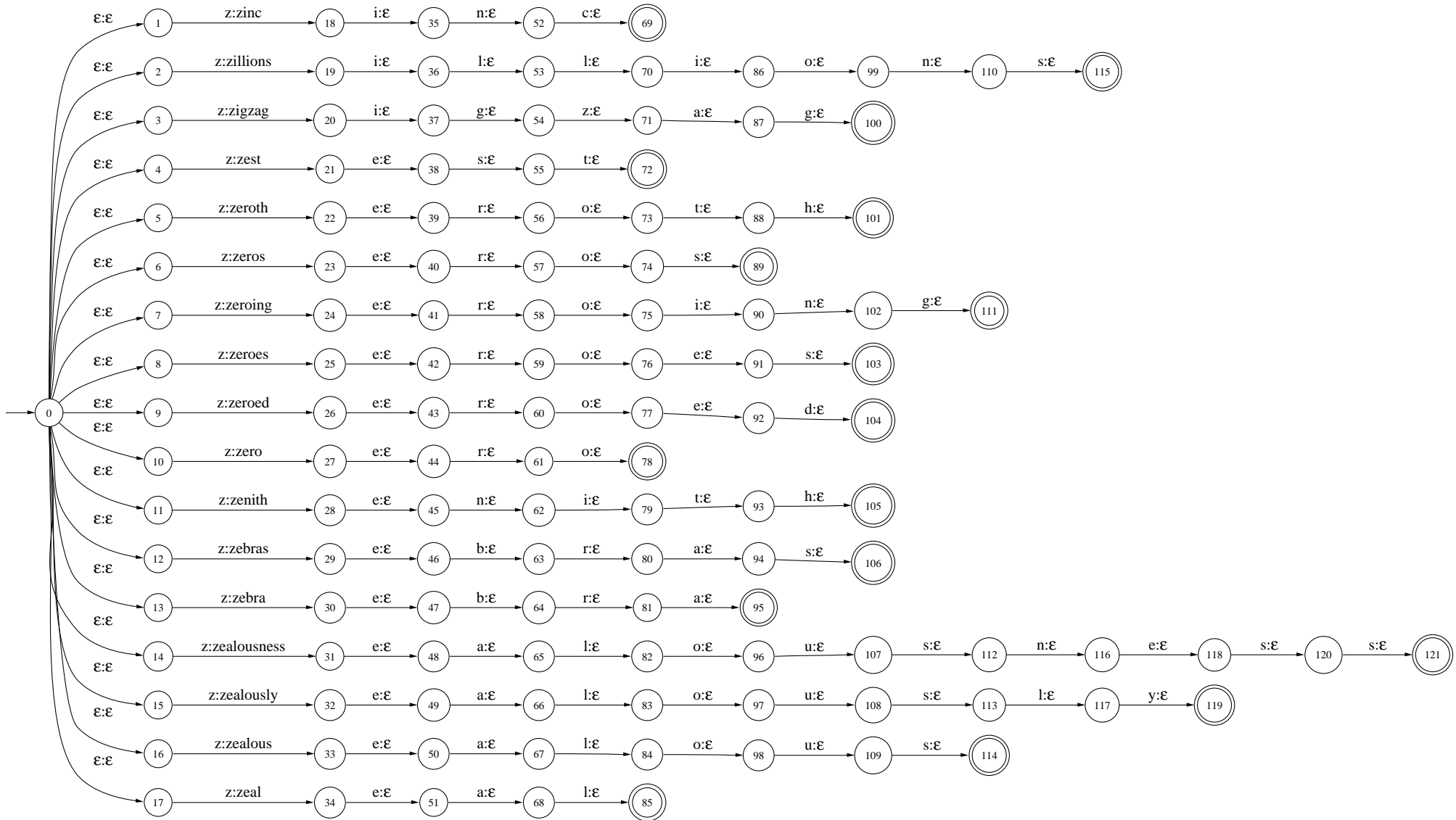
Minimization (Identity)

- minimal \neq minimal number of states
- minimal \equiv deterministic with minimal number of states
- merge *equivalent* states, will not increase size
- cyclic $O(N \log N)$, acyclic $O(N)$

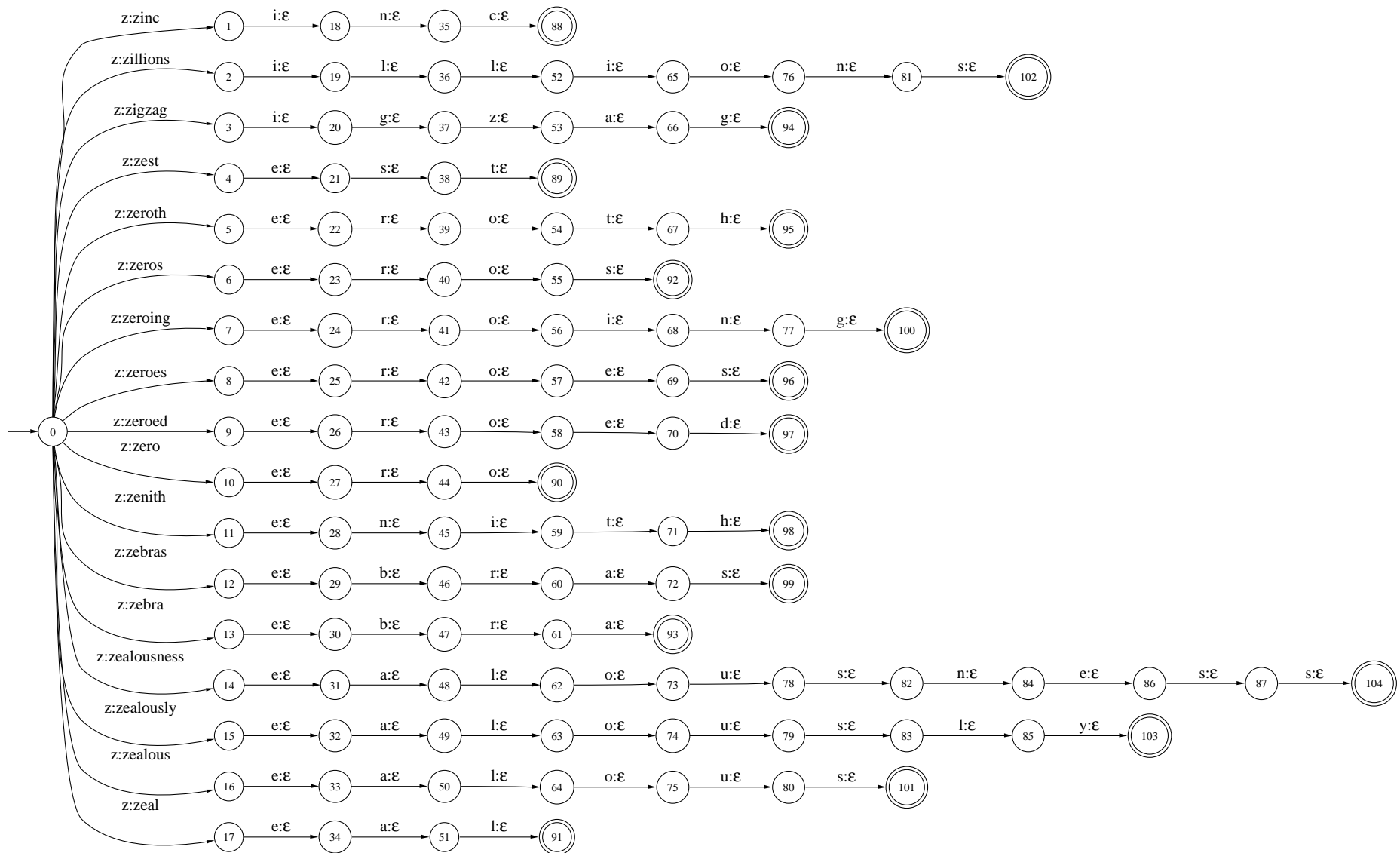




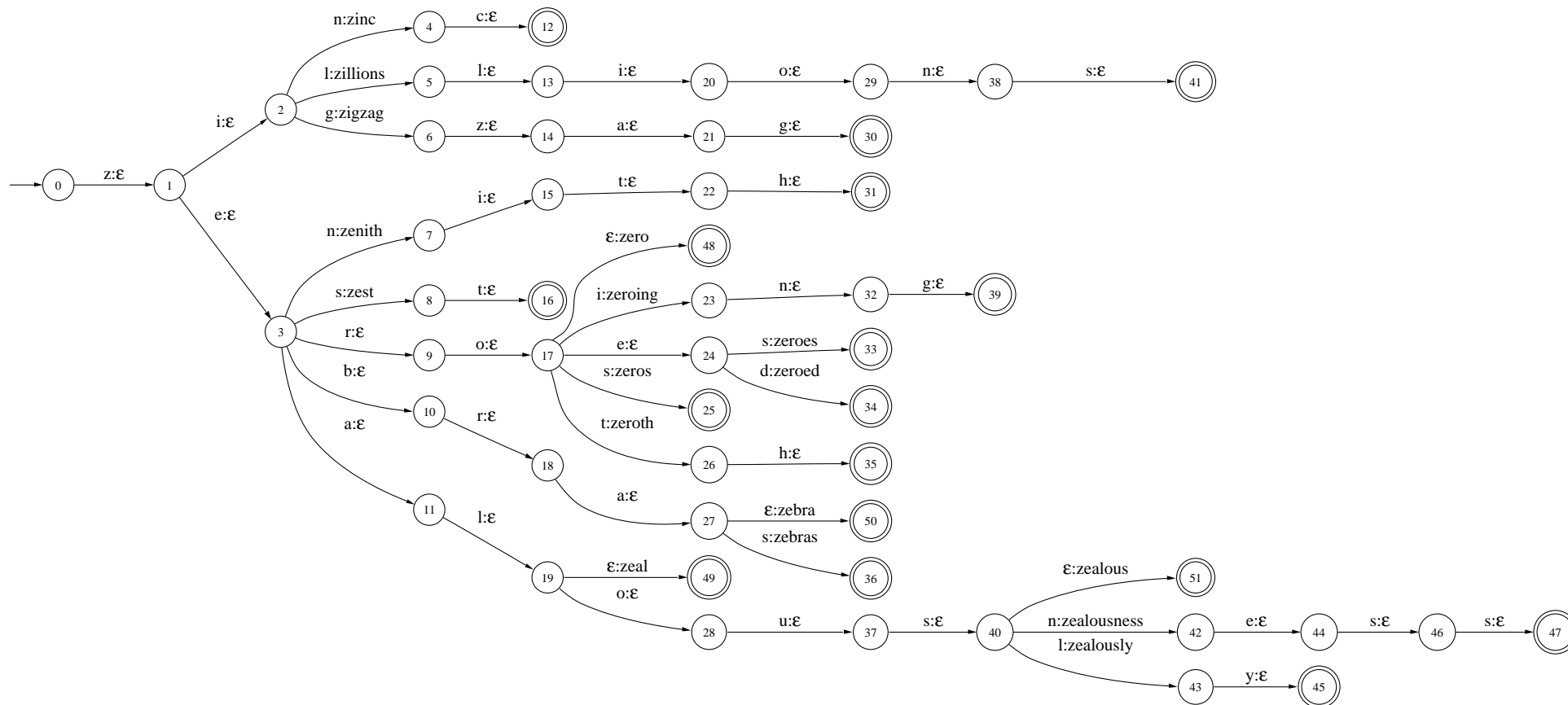
Example Lexicon



Example Lexicon: ϵ Removed

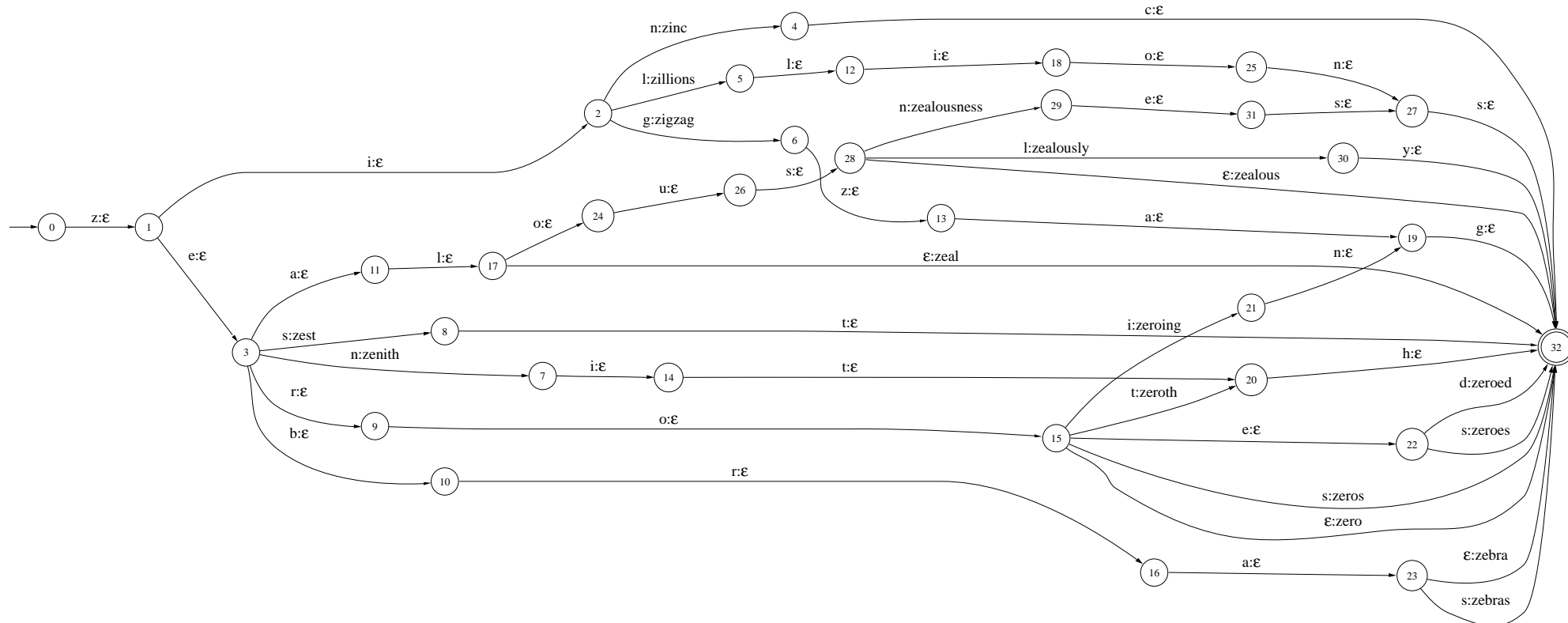


Example Lexicon: Determinized



- lexical tree
- sharing at beginning of words: can prune many words at once

Example Lexicon: Minimized



- sharing at the end of words

On-The-Fly Implementation

- lazy evaluation: generate only relevant states/transitions
- enables use of infinite-state machines (e.g., CFG)
- on-the-fly:
 - composition, intersection
 - union, concatenation, closure
 - ϵ removal, determinization
- not on-the-fly:
 - trimming dead states
 - minimization
 - reverse

FSTs in Speech Recognition

- cascade of FSTs:

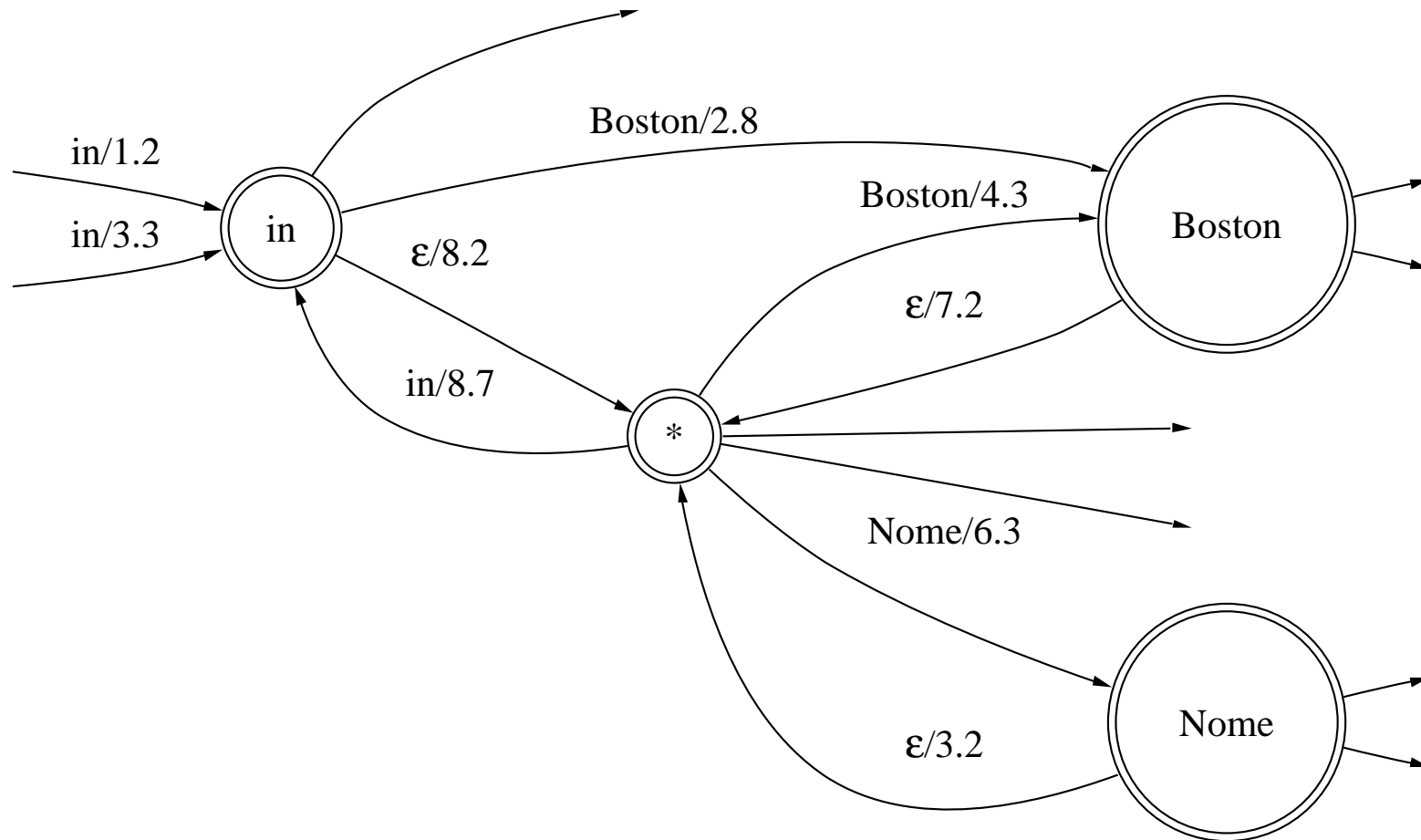
$$(S \circ A) \circ \underbrace{(C \circ P \circ L \circ G)}_R$$

- S : acoustic segmentation*
- A : application of acoustic models*
- C : context-dependent relabeling (e.g., diphones, triphones)
- P : phonological rules
- L : lexicon
- G : grammar/language model (e.g., n -gram, finite-state, RTN)

FSTs in Speech Recognition

- in practice:
 - $S \circ A$ is acoustic segmentation with on-demand model scoring
 - $C \circ P \circ L \circ G$: precomputed and optimized or expanded on the fly
 - composition $S \circ A$ with $C \circ P \circ L \circ G$ computed on demand during forward Viterbi search
 - might use multiple passes, perhaps with different G
- advantages:
 - forward search sees a *single* FST $R = C \circ P \circ L \circ G$, doesn't need special code for language models, lexical tree copying, etc. . .
 - can be very fast
 - easy to do cross-word context-dependent models

N -gram Language Model (G): Bigram



- each distinct word history has its own state
- direct transitions for each existing n -gram
- ϵ transitions to back-off state (*),
 ϵ removal undesirable

Phonological Rules (P)

- segmental system needs to match explicit segments
- ordered rules of the form:

$$\{V \text{ SV}\} \quad b \quad \{V \text{ l r w}\} \quad \Rightarrow \quad bc1 \quad [b] \quad ;$$

$$\{m\} \quad b \quad \{\} \quad \Rightarrow \quad [bc1] \quad b \quad ;$$

$$\{\} \quad b \quad \{\} \quad \Rightarrow \quad bc1 \quad b \quad ;$$

$$\{s\} \quad s \quad \{\} \quad \Rightarrow \quad [s] \quad ;$$

$$\{\} \quad s \quad \{\} \quad \Rightarrow \quad s \quad ;$$

- rule selection deterministic, rule replacement may be ambiguous
- compile rules into transducer $P = P_l \circ P_r$
 - P_l applied left-to-right
 - P_r applied right-to-left

MIT

EM Training of FST Weights

- FSA A_x given set of examples x
 - straightforward application of EM to train $P(x)$
 - our tools can also train an RTN (CFG)
- FST $T_{x:y}$ given set of example pairs $x : y$
 - straightforward application of EM to train $T_{x,y} \Rightarrow P(x, y)$
 - $T_{x|y} = T_{x,y} \circ [\det(T_y)]^{-1} \Rightarrow P(x|y)$ [Bayes' Rule]
- FST $T_{x|y}$ within cascade $S_{v|x} \circ T_{x|y} \circ U_z$ given $v : z$
 - $x = v \circ S$
 - $y = U \circ z$
 - train $T_{x|y}$ given $x : y$
- We have used these techniques to train P , L , and $(P \circ L)$.

MIT

Conclusion

- introduced FSTs and their basic operations
- use of FSTs throughout system adds consistency and flexibility
- consistency enables powerful algorithms everywhere (write algorithms once)
- flexibility enables new and unforeseen capabilities (but enables you to hang yourself too)
- SUMMIT (Jupiter) 25% faster when converted to FST framework, yet much more flexible

- E. Roche and Y. Schabes (eds.), *Finite-State Language Processing*, MIT Press, Cambridge, 1997.
- M. Mohri, “Finite-state transducers in language and speech processing,” in *Computational Linguistics*, vol. 23, 1997.
- M. Mohri, M. Riley, D. Hindle, A. Ljolje, F. Pereira, “Full expansion of context-dependent networks in large vocabulary speech recognition”, in Proc. ICASSP, Seattle, 1998.