# Language and Statistics II

## Lecture 8: Applications and Learning of WFSTs

Noah Smith

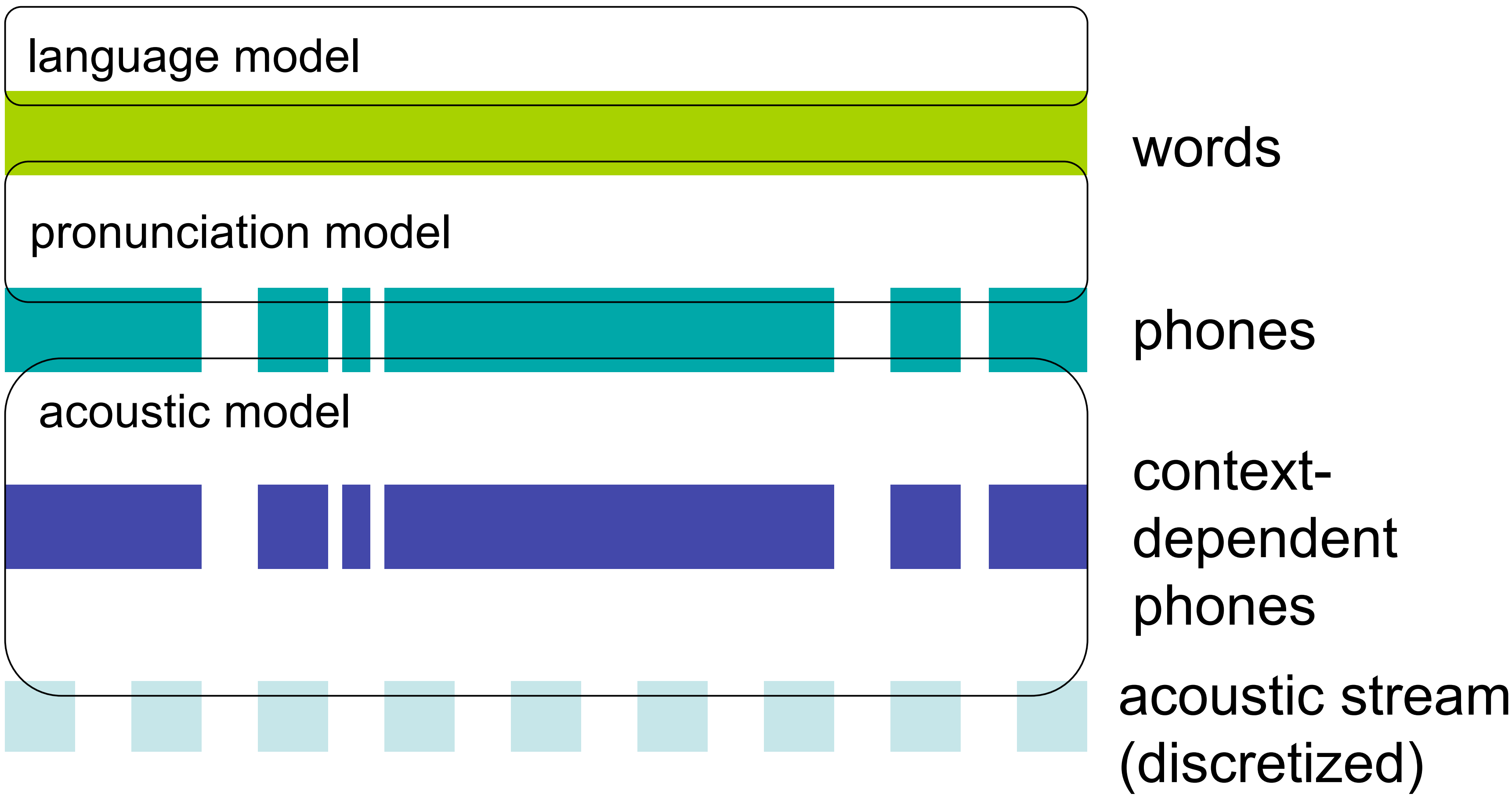
# Lecture Outline

- WFSTs in speech recognition
- WFSTs for machine translation
- Semirings
- Parameter estimation for WFSTs

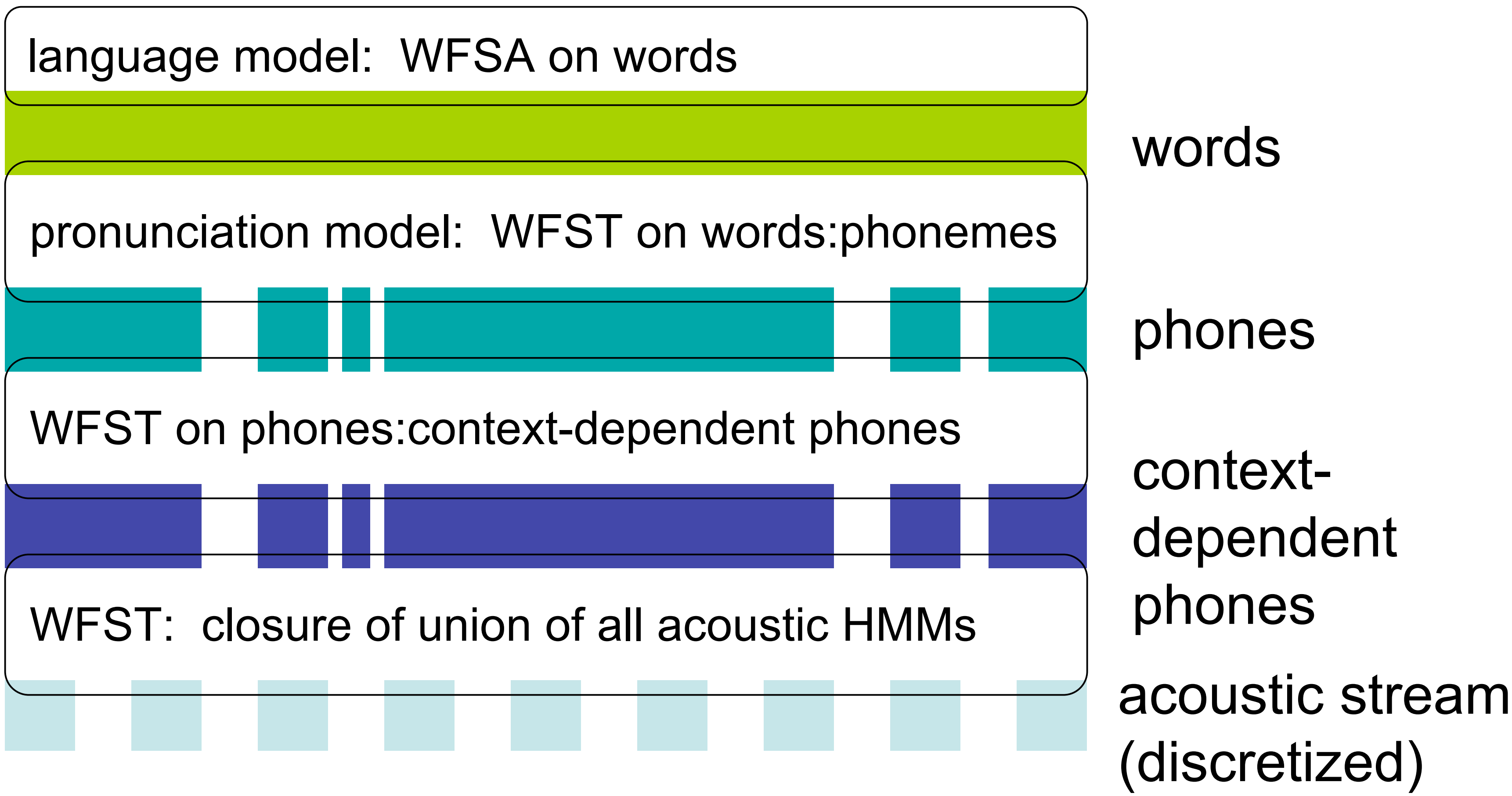
  part of Eisner (2002)
- Grammatical inference for finite-state models

  Stolcke and Omohundro (1993)

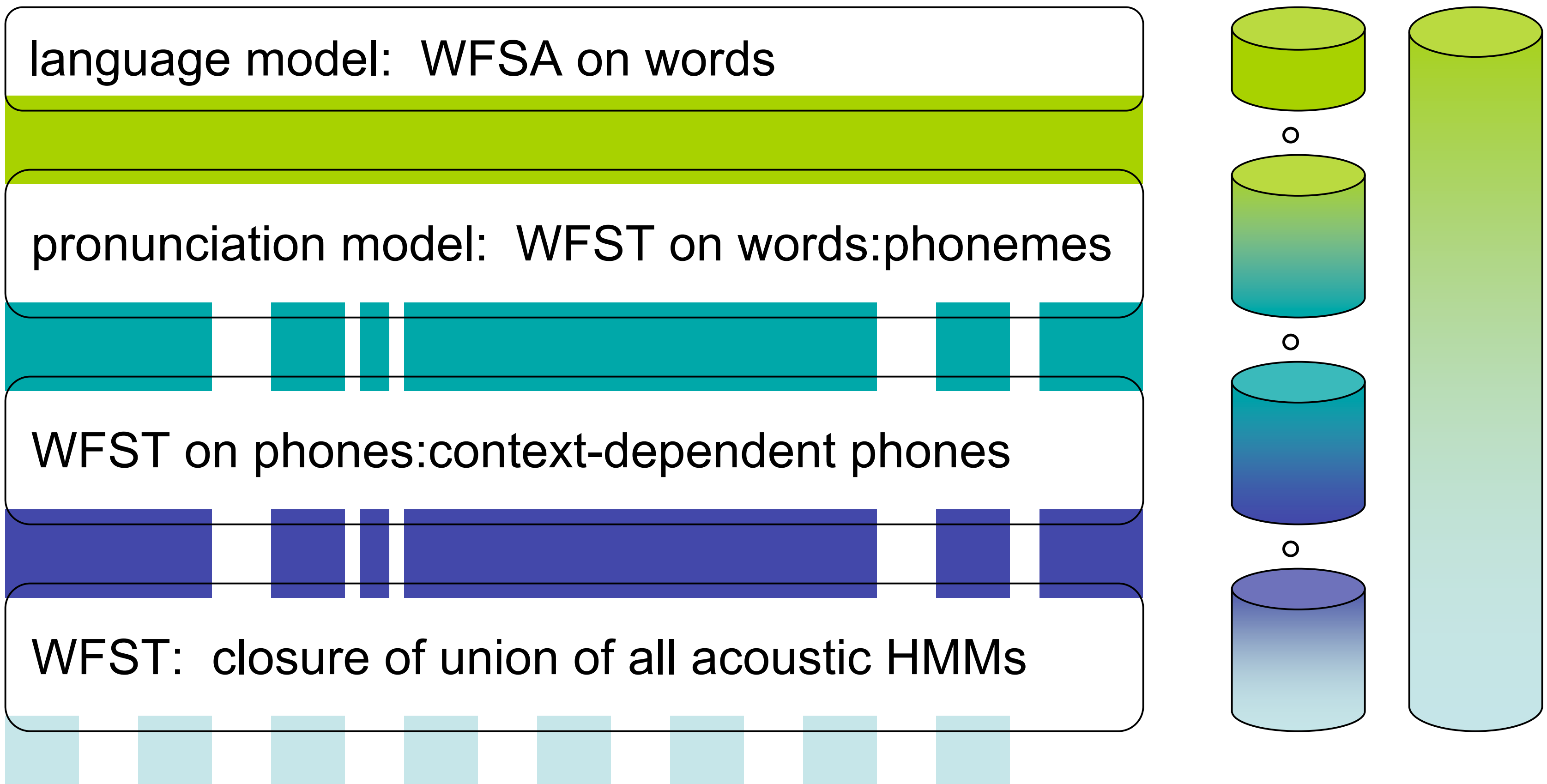
# From Acoustics to Text
## (Mohri & Riley)



# From Acoustics to Text
## (Mohri & Riley)



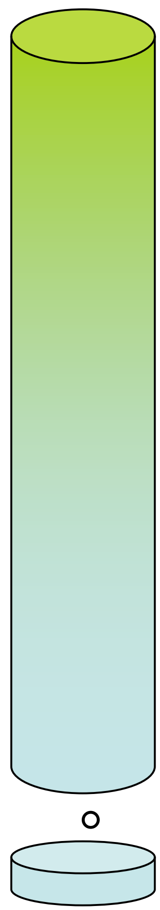
# From Acoustics to Text
## (Mohri & Riley)



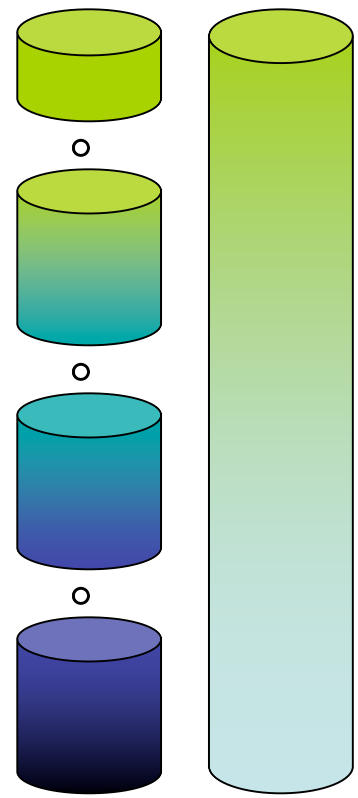
# From Acoustics to Text
## (Mohri & Riley)

At runtime: given acoustic sequence **as input**, find the **best path** through the WFST.

In principle, could **compose** and get FSA of hypotheses …



All of this is done **offline**.

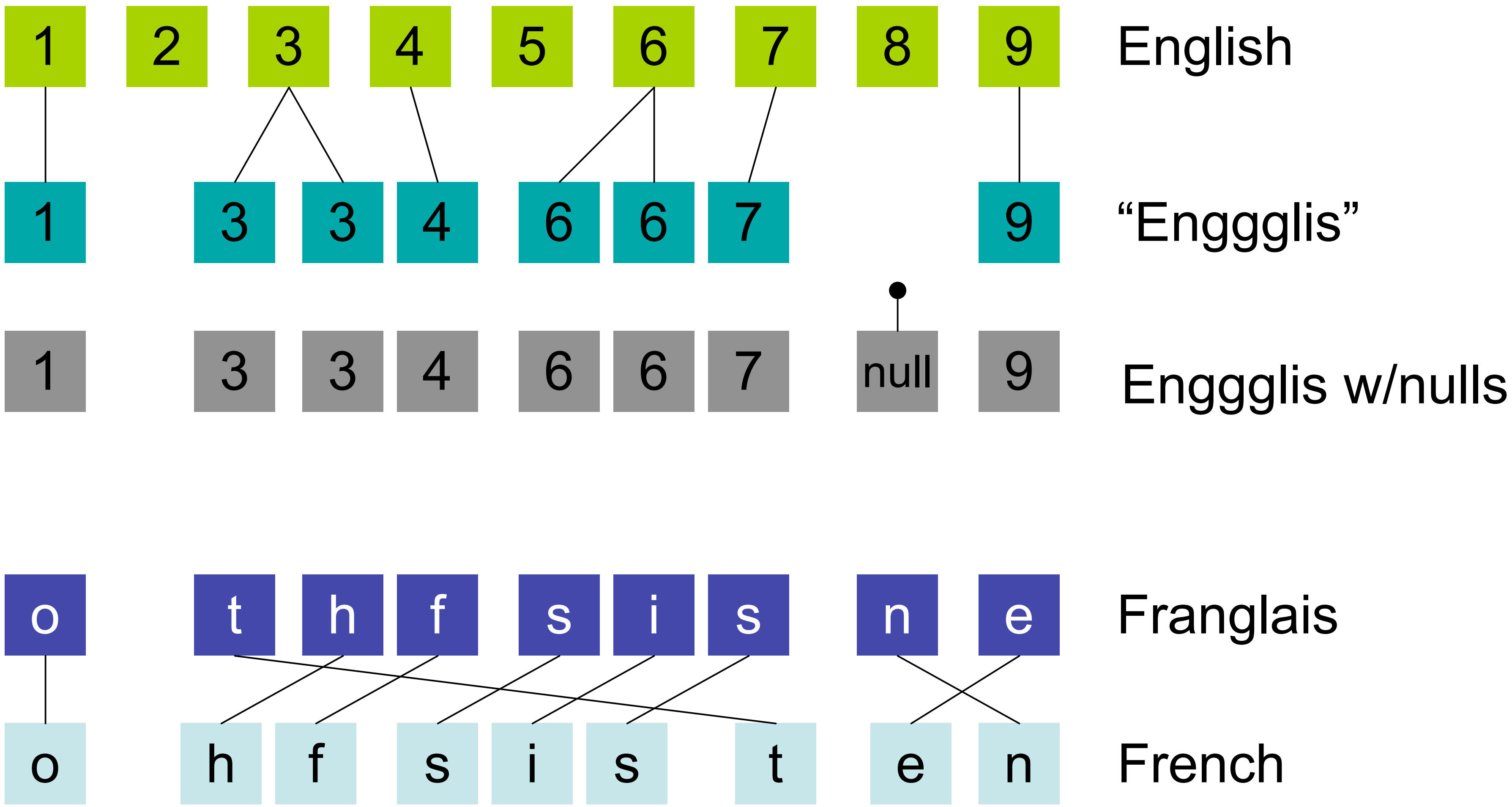


# From French to English
(Knight and Al-Onaizan)

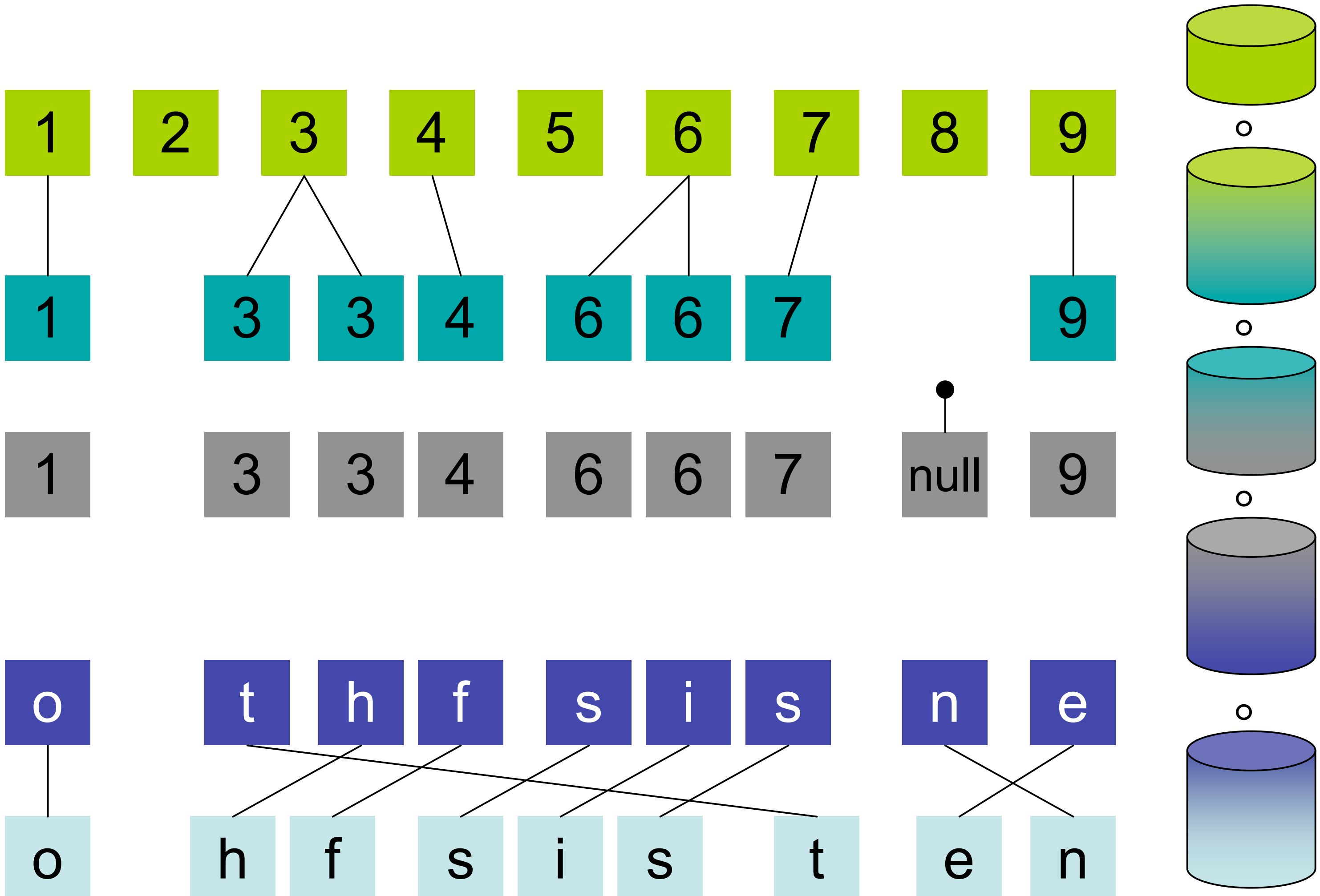
- IBM model 3: give the probability of a sequence of French words **given** a sequence of English words: p(F | E) (translation model)
- Combine with language model, p(E).
- These are very simple models, but exact decoding is rather hard. (NP hard in fact.)

$$\text{translate}(F) = \text{argmax}_E \; p(E, F)$$

# From French to English
## (IBM Model 3)



# From French to English
## (Knight and Al-Onaizan; Model 3 with WFSTs)



# From French to English
## (Knight and Al-Onaizan; Model 3 with WFSTs)

WFSA English language model

∘

WFST fertility model

∘

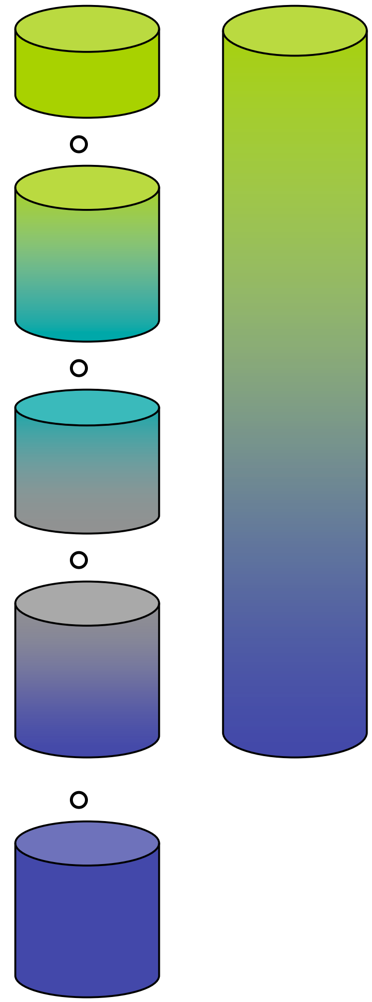
WFST null insertion model

∘

WFST word translation model

∘

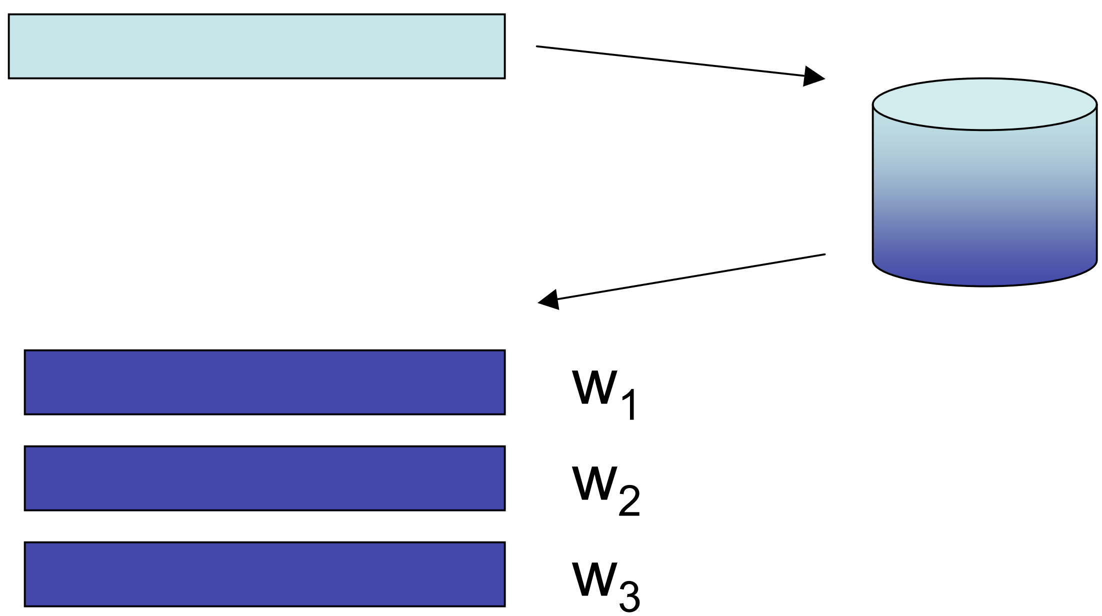
(W)FSA of all French permutations



Back to more general discussion of WFSTs.

# What do WFSTs encode?

- Weighted relations on strings.



# When weights have a probability interpretation …

- The weight of a **path** is the **product** of all the arcs' weights in the path.
- The weight of an output **string** (given an input string) is the **sum** of path weights.

Alternative:

- If we want the **best** path, use **max** instead of sum.

*Weights can have different interpretations.*

# Semirings

| *Interpretation of weights* | *Want to compute* | *weights* | *“plus”* | *“times”* |
|---|---|---|---|---|
| probabilities | p(s) | [0, 1] | + | × |
| | best path prob. | | max | × |
| log-probabilities | log p(s) | (-∞, 0] | log+ | + |
| | best path log-prob. | | max | + |
| costs | min-cost path | [0, +∞) | min | + |
| Boolean | s in language? | {0, 1} | OR | AND |
| strings | s itself | Σ* | set-union | concat. |

# Weighted Composition

- What does weighted composition **mean**?
- The semantics we want, given the models we've been looking at,

$$\underbrace{p_C(z|x)}_{\substack{\text{new composed transducer;}\\ C = A \circ B}} = \sum_y p_B(z|y) \times p_A(y|x)$$

- The point: this is specific to the **semiring**. If we had the Boolean semiring, we'd use OR and AND … and get old-fashioned intersection!

# Application of Weighted Composition: Path Sum

- In the probability case, build x ° T ° y … result is a WFST encoding all of the ways (paths) to recognize (x, y).
- Sum up **weights** of those paths = path sum.
  - One way to do it: replace all input and output symbols with ε; project, determinize, minimize … end up with a single state FSM whose weight is the path sum.
  - Another way: convert to a linear system.
- Generalizes to other semirings.
  - max path, existence of path, etc.
- Special case: **forward** algorithm's trellis = big composed machine!
- Special case: **Viterbi** is the same thing, in max/× semiring.

# Why are Path Sums Important?

- Total weight of all the paths that **meet some constraints**, such as:
  - match input
  - match output
  - match both input and output

# Why are Path Sums Important?

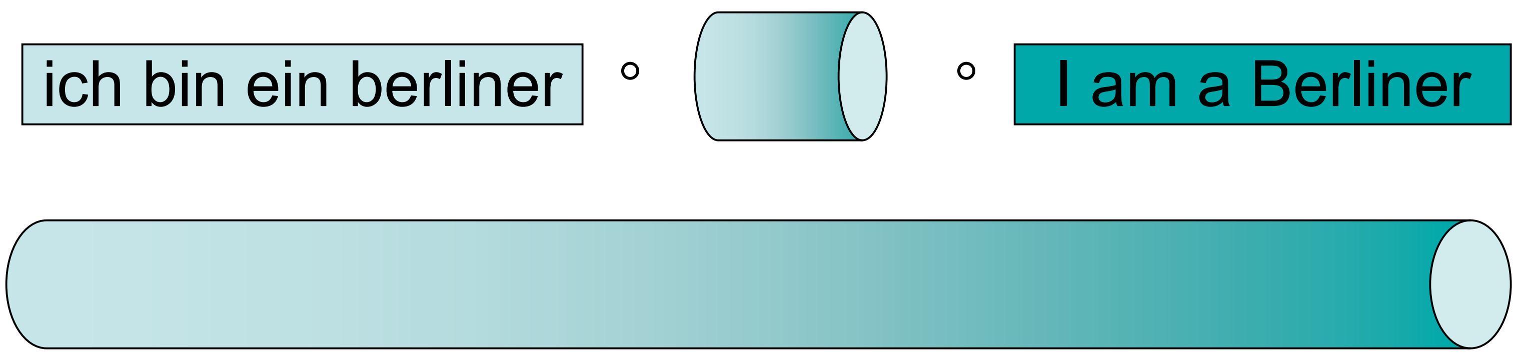
- Total weight of all the paths that **meet some constraints**, such as:
  - match input   $p(x)$
  - match output   $p(y)$
  - match both input and output   $p(x, y)$ or $p(y \mid x)$

# Where do the weights come from?

- In an HMM or n-gram model, if you have **annotated data**, you can use it to estimate arc probabilities. (MLE, perhaps smoothed.)
- Generalization to WFSTs: if we have a bunch of **observed paths**, we can do the same thing.
- What if we only have inputs & outputs (no paths)?

# Suppose we **don't** have paths.

- For today: assume you **do** have the input & the output …




path sum = p(ich bin ein berliner, I am a Berliner)

# Training Weighted FSTs
## (Eisner, 2002)

- p(x, y) … likelihood of the data under the model.
- Reminiscent of other log-linear models … the **data** and the model class (**features**) define a function (**likelihood**) that we want to maximize.
- General point: training weights usually means defining an iterative **update rule**, e.g., based on gradients.
- Can we compute the **gradient**?

# Training Weighted FSTs
## (Eisner, 2002)

- Eisner's solution: include gradient as *part of the weight*.
- Before: weight is a probability.
- Now: weight is (probability, gradient).

- Key idea: this (p, $\nabla$) weight is a **semiring**!

# Semirings

| *Interpretation of weights* | *Want to compute* | *weights* | *"plus"* | *"times"* |
|---|---|---|---|---|
| probabilities | p(s) | [0, 1] | + | × |
| probability, gradient | p(s), $\nabla_w p(s)$ | $[0, 1] \times \mathbb{R}^E$ | $(p_1, \mathbf{g}_1) \oplus (p_2, \mathbf{g}_2)$ = $(p_1 + p_2, \mathbf{g}_1 + \mathbf{g}_2)$ | $(p_1, \mathbf{g}_1) \otimes (p_2, \mathbf{g}_2)$ = $(p_1 p_2, p_2\mathbf{g}_1 + p_1\mathbf{g}_2)$ |

# Training Weighted FSTs
## (Eisner, 2002)

- Generalization: if the output weren't known, the whole thing goes through the same way … just replace y with an FSM that accepts anything!
  - Same if we have "noisy" y.
- Generalization: if arcs have multiple features, modify their vectors to be feature count vectors, instead of [0 … 1 … 0].
  - "Parameterized" WFSTs.
- The "gradients" have another interpretation …. expectations of the number of times we cross each arc. This will come in handy later.

# Training Weighted FSTs
## (Eisner, 2002)

So the training method …

1. Initialize the *i*th arc in T with weights for T with $(w_i, [0\ 0\ 0\ \ldots\ 1\ \ldots\ 0])$; *i*th cell is 1
2. For each example ($x$, $y$), build $x \circ T \circ y$.
3. Compute the path sum in the expectation semiring. This gives $p(x, y)$ and $\nabla_{\mathbf{w}} p(x, y)$.
4. Update: $\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} p(x, y)$
5. If not converged, go to 2.

Swept under the rug: making sure the weights are well-formed arc probabilities. Also efficiency.