

Part I: Designing HMM-based ASR systems
Part II: Training continuous density HMMs

Rita Singh

School of Computer Science
Carnegie Mellon University

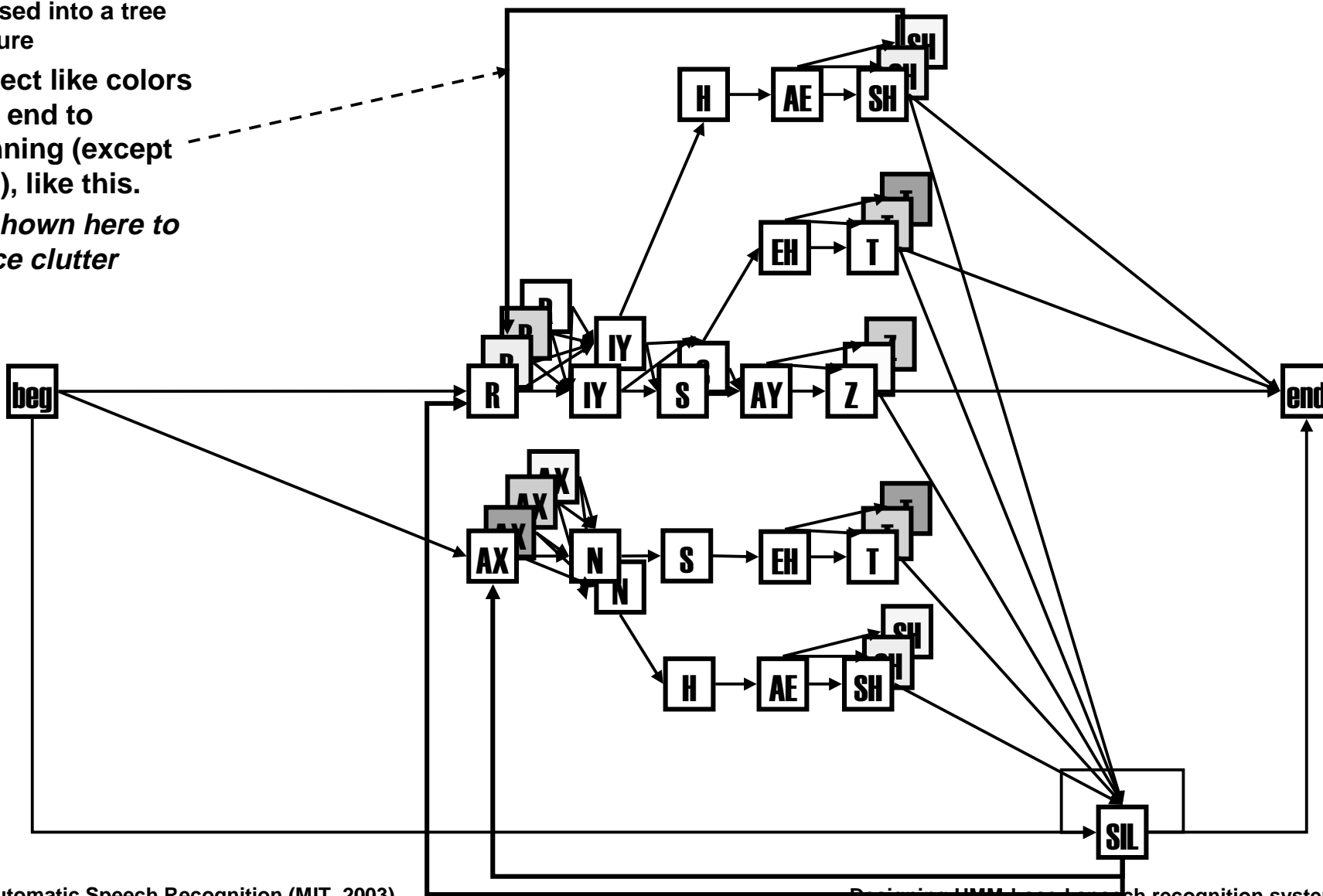
Designing graph structures for the language
HMM : Understanding and dealing with the
complexity introduced by the use of sub-word
units

Building the triphone-based UNIGRAM *LEXTREE* sentence HMM

In a lextree, phones are collapsed into a tree structure

Connect like colors from, end to beginning (except white), like this.

Not shown here to reduce clutter

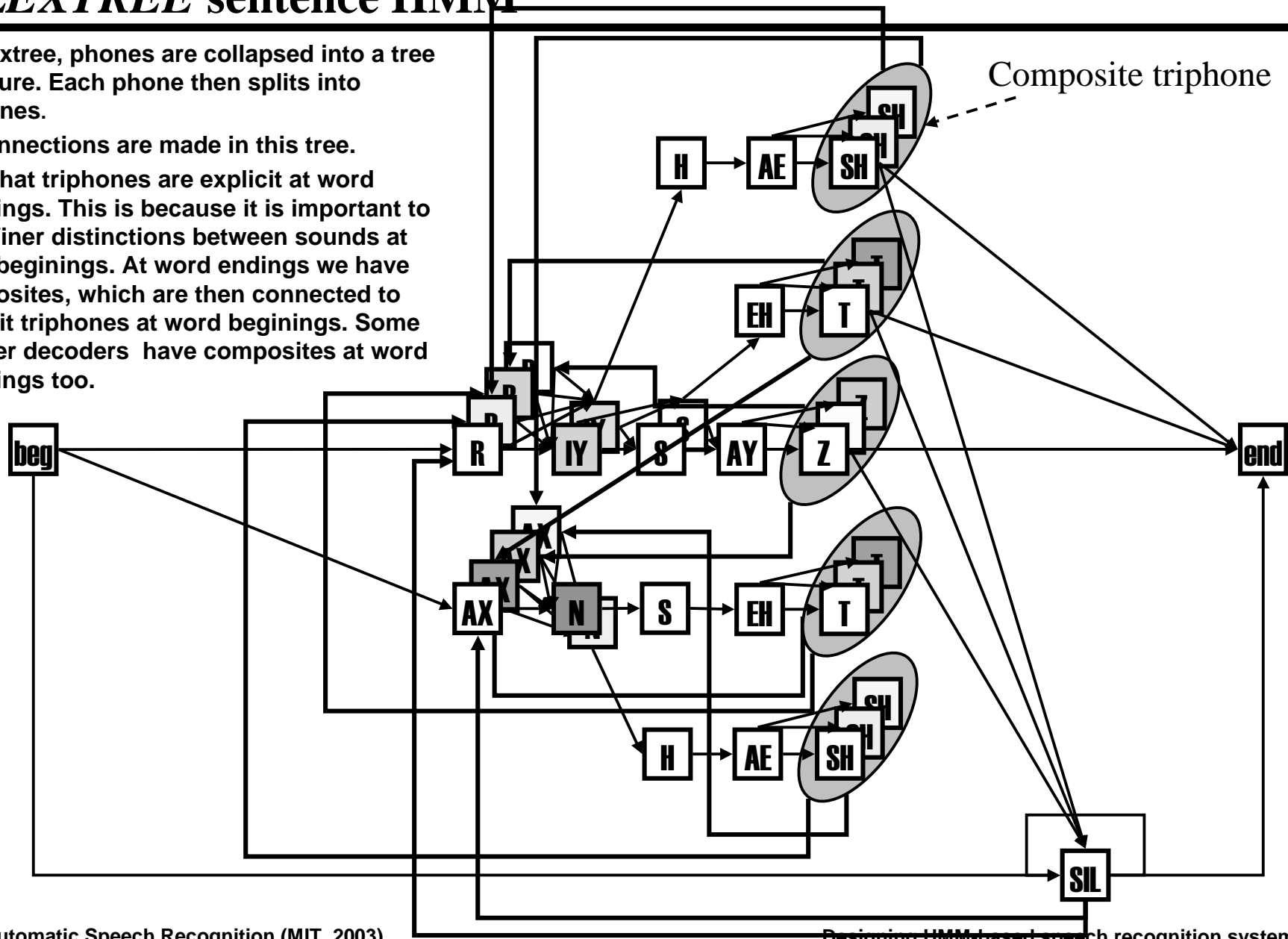


Building the triphone-based UNIGRAM *LEXTREE* sentence HMM

In a lextree, phones are collapsed into a tree structure. Each phone then splits into triphones.

All connections are made in this tree.

Note that triphones are explicit at word beginnings. This is because it is important to have finer distinctions between sounds at word beginnings. At word endings we have composites, which are then connected to explicit triphones at word beginnings. Some simpler decoders have composites at word beginnings too.



PART II

Training continuous density HMMs

Baum-Welch: Computing *A Posteriori* State Probabilities and Other Counts

- ◆ Compute α and β terms using the forward backward algorithm

$$\alpha(s, t | word) = \sum_{s'} \alpha(s', t-1 | word) P(s | s') P(X_t | s)$$

$$\beta(s, t | word) = \sum_{s'} \beta(s', t+1 | word) P(s' | s) P(X_{t+1} | s')$$

- ◆ Compute *a posteriori* probabilities of states and state transitions using α and β values

$$\gamma(s, t | word) = \frac{\alpha(s, t) \beta(s, t)}{\sum_{s'} \alpha(s', t) \beta(s', t)}$$

$$\gamma(s, t, \tilde{s}, t+1 | word) = \frac{\alpha(s, t) P(\tilde{s} | s) P(X_{t+1} | \tilde{s}) \beta(\tilde{s}, t+1)}{\sum_{s'} \alpha(s', t) \beta(s', t)}$$

