A TAG-based noisy channel model of speech repairs

Mark Johnson and Eugene Charniak

Brown University

ACL, 2004

Supported by NSF grants LIS 9720368 and IIS0095940

Talk outline

- Goal: Apply parsing technology and "deeper" linguistic analysis to (transcribed) speech
- Problem: Spoken language contains a wide variety of *disfluencies* and *speech errors*
- Why speech repairs are problematic for statistical syntactic models
  - Statistical syntactic models capture *nested head-to-head dependencies*
  - Speech repairs involve *crossing "rough-copy" dependencies* between sequences of words
- A noisy channel model of speech repairs
  - Source model captures syntactic dependencies
  - Channel model introduces speech repairs
  - *Tree adjoining grammar* can formalize the non-CFG dependencies in speech repairs

Speech errors in (transcribed) speech

- **Filled pauses**

  I think it's, *uh*, refreshing to see the, *uh*, support ...

- **Parentheticals**

  But, *you know*, I was reading the other day ...

- **Speech repairs**

  Why didn't he, why didn't she stay at home?

- **"Ungrammatical" constructions, i.e., non-standard English**

  My friends is visiting me?

  (Note: this really isn't a speech error)

Bear, Dowding and Schriberg (1992), Charniak and Johnson (2001), Heeman and Allen (1997, 1999), Nakatani and Hirschberg (1994), Stolcke and Schriberg (1996)

Special treatment of speech repairs

- *Filled pauses* are easy to recognize (in transcripts)
- *Parentheticals* appear in our training data and our parsers identify them fairly well
- *Filled pauses* and *parentheticals* are useful for identifying constituent boundaries (just as punctuation is)
  - Our parser performs slightly better with parentheticals and filled pauses than with them removed
- *"Ungrammaticality" and non-standard English* aren't necessarily fatal
  - Statistical parsers learn how to map sentences to their parses from a training corpus
- ... but *speech repairs* warrant special treatment, since our parser never recognizes them even though they appear in the training data ...

Engel, Charniak and Johnson (2002) "Parsing and Disfluency Placement", EMNLP

The structure of speech repairs

$$\ldots\text{a flight } \underbrace{\text{to Boston,}}_{\text{Reparandum}}\ \underbrace{\text{uh, I mean,}}_{\text{Interregnum}}\ \underbrace{\text{to Denver}}_{\text{Repair}}\text{ on Friday }\ldots$$

- The Interregnum is usually lexically (and prosodically marked), but can be empty
- Repairs don't respect syntactic structure

  Why didn't she, uh, why didn't he stay at home?

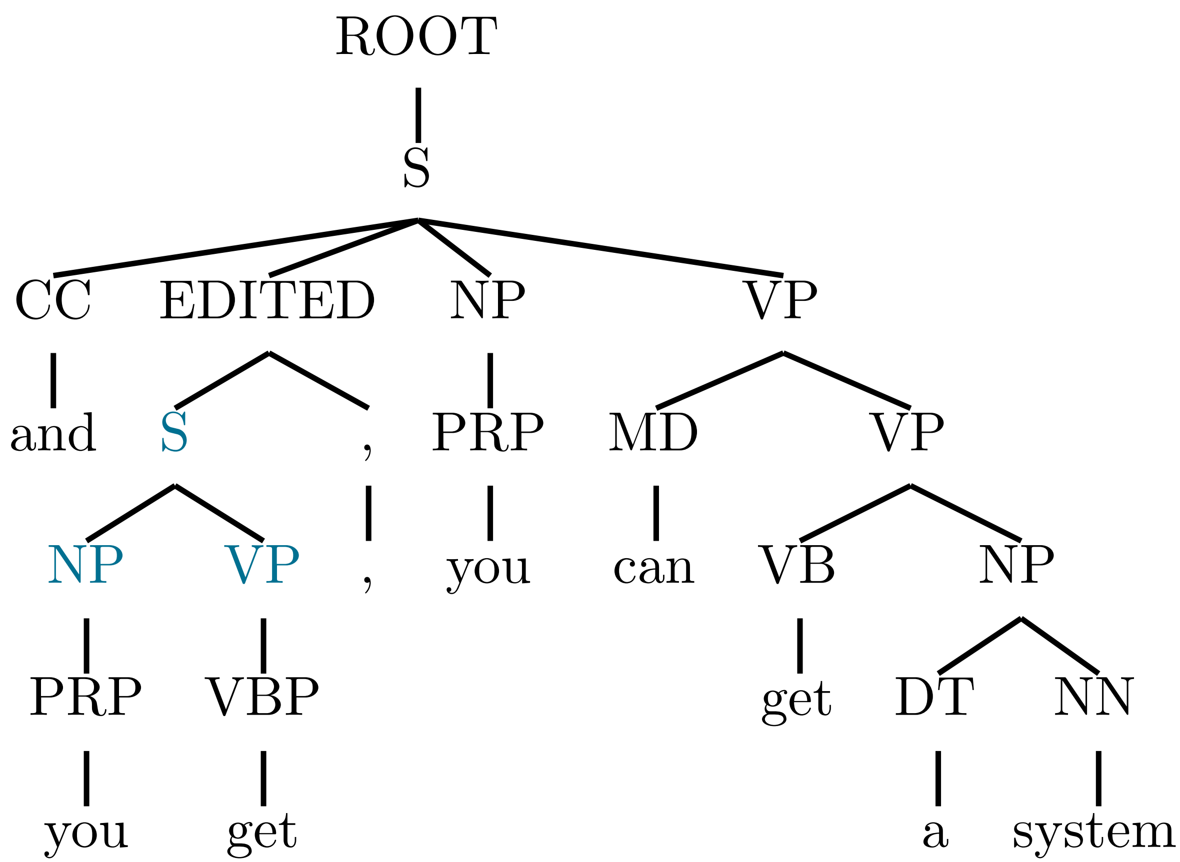
- *The Repair is often "roughly" a copy of the Reparandum*

  ⇒ identify repairs by looking for "rough copies"

- The Reparandum is often 1–2 words long (⇒ word-by-word classifier)
- The Reparandum and Repair can be completely unrelated

Shriberg (1994) "Preliminaries to a Theory of Speech Disfluencies"

Representation of repairs in treebank



- Speech repairs are indicated by EDITED nodes in corpus
- The internal syntactic structure of EDITED nodes is highly unusual

Speech repairs and interpretation

- Speech repairs are indicated by EDITED nodes in corpus
- The parser does not posit any EDITED nodes even though the training corpus contains them
  - Parser is based on context-free headed trees and head-to-argument dependencies
  - Repairs involve *rough copy* dependencies that cross constituent boundaries

Why didn't he, uh, why didn't she stay at home?

  - Finite state and context free grammars cannot generate ww "copy languages" (*but Tree Adjoining Grammars can*)
- The interpretation of a sentence with a speech repair is (usually) the same as with the repair excised

⇒ *Identify and remove EDITED words before parsing*
  - Use a classifier to classify each word as "EDITED" or "not EDITED" (Charniak and Johnson, 2001)
  - Use a *noisy channel model* to generate/remove repairs

The noisy channel model

Source model $\mathrm{P}(X)$
Bigram/Parsing LM

Source signal x
a flight to Denver on Friday

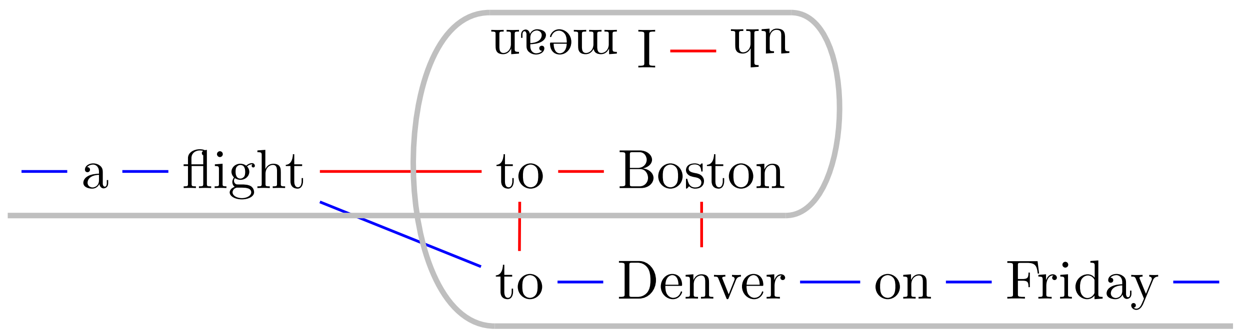
Noisy channel $\mathrm{P}(U|X)$
TAG transducer

Noisy signal u
a flight to Boston uh I mean to Denver on Friday

- $\mathrm{argmax}_x \, \mathrm{P}(x|u) \;=\; \mathrm{argmax}_x \, \mathrm{P}(u|x)\mathrm{P}(x)$
- Train source language model on treebank trees *with EDITED nodes removed*

“Helical structure” of speech repairs

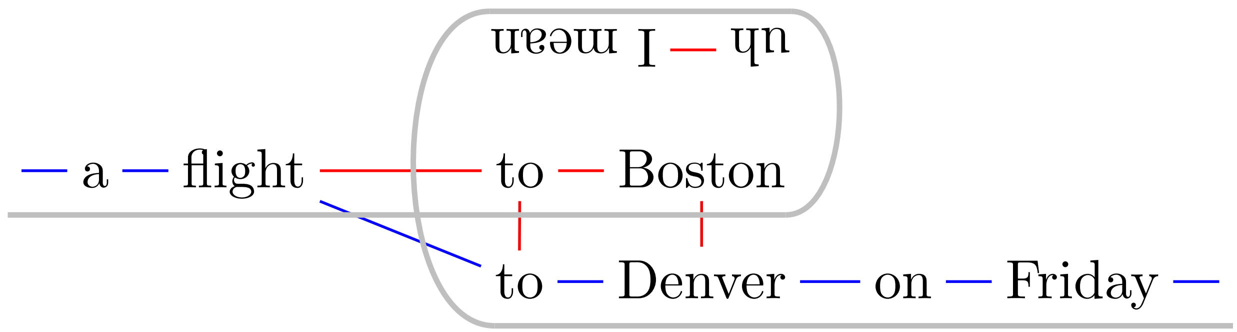
. . . a flight $\underbrace{\text{to Boston,}}_{\text{Reparandum}}$ $\underbrace{\text{uh, I mean,}}_{\text{Interregnum}}$ $\underbrace{\text{to Denver}}_{\text{Repair}}$ on Friday . . .



- *Parser-based language model* generates *repaired string*
- *TAG transducer* generates *reparandum* from repair
- *Interregnum* is generated by specialized finite state grammar in TAG transducer

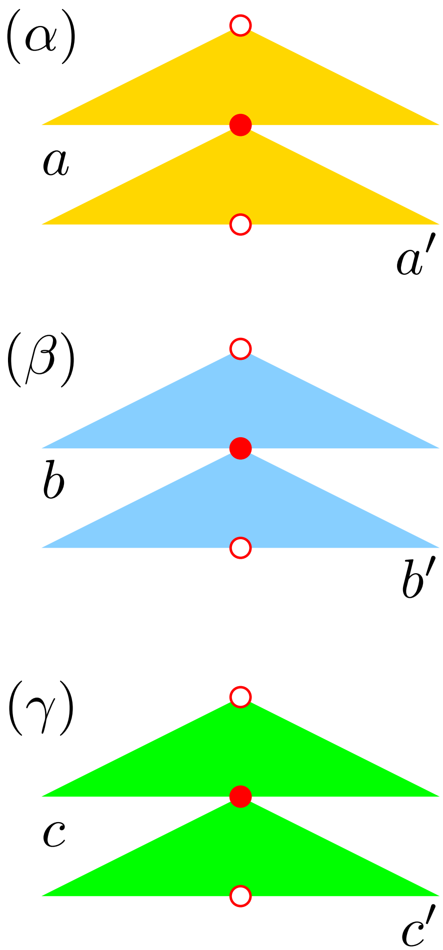
Joshi (2002), ACL Lifetime achievement award talk

TAG transducer models speech repairs

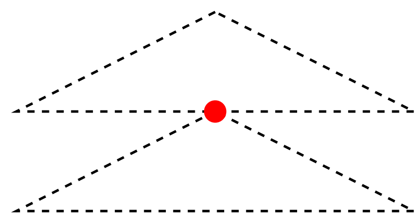


- Source language model: *a flight to Denver on Friday*
- TAG generates string of $u{:}x$ pairs, where u is a speech stream word and x is either $\emptyset$ or a source word:
  a:a flight:flight to:$\emptyset$ *Boston:*$\emptyset$ *uh:*$\emptyset$ *I:*$\emptyset$ *mean:*$\emptyset$ *to:to Denver:Denver on:on Friday:Friday*
  - TAG does not reflect grammatical structure (the LM does)
  - *right branching finite state* model of *non-repairs* and *interregnum*
  - *TAG adjunction* used to describe *copy dependencies in repair*

TAG derivation of copy constructions



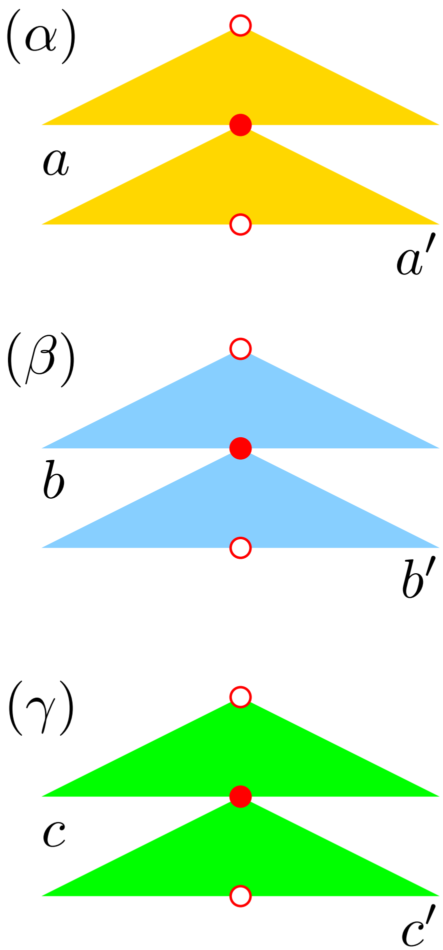
Auxiliary trees



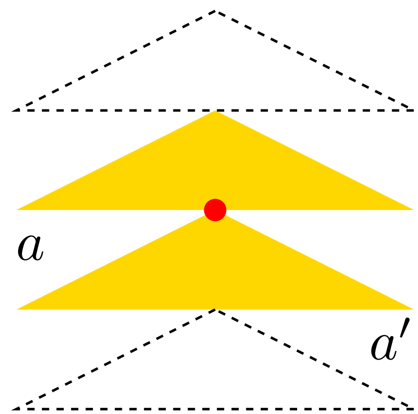
Derived tree

Derivation tree

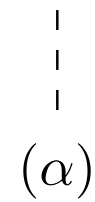
TAG derivation of copy constructions



Auxiliary trees

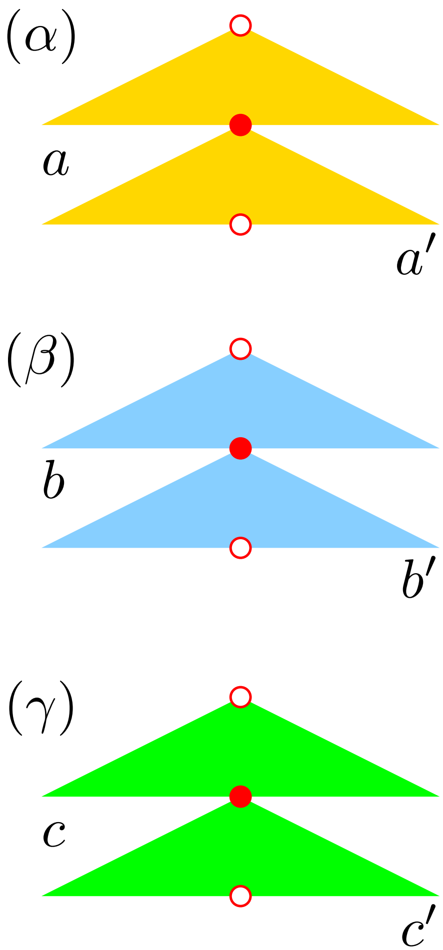


Derived tree

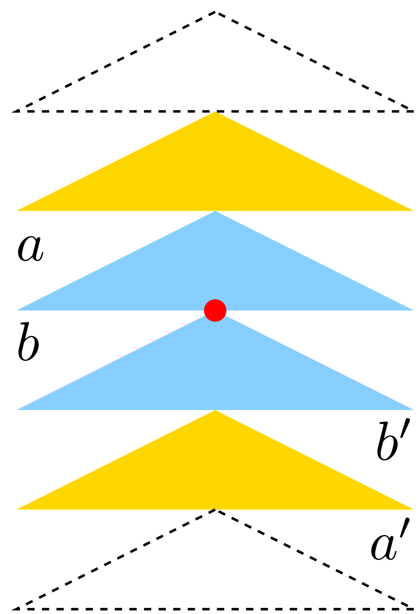


Derivation tree

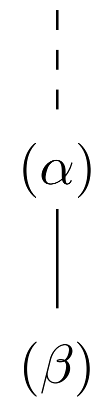
TAG derivation of copy constructions



Auxiliary trees

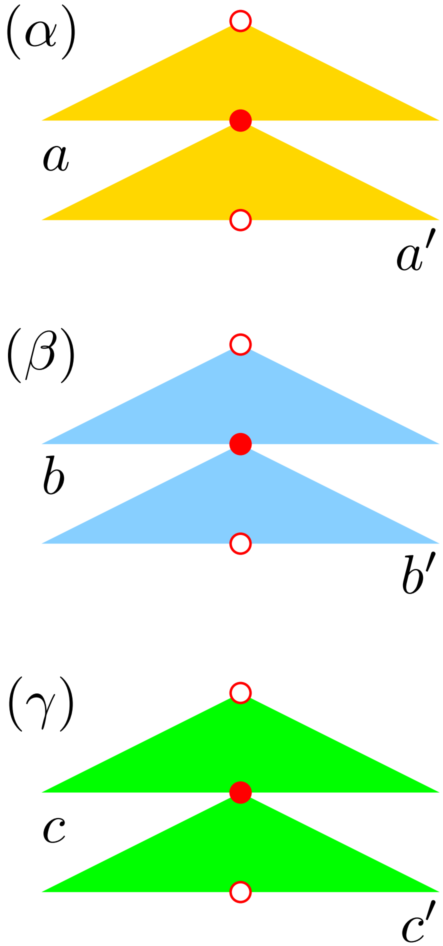


Derived tree

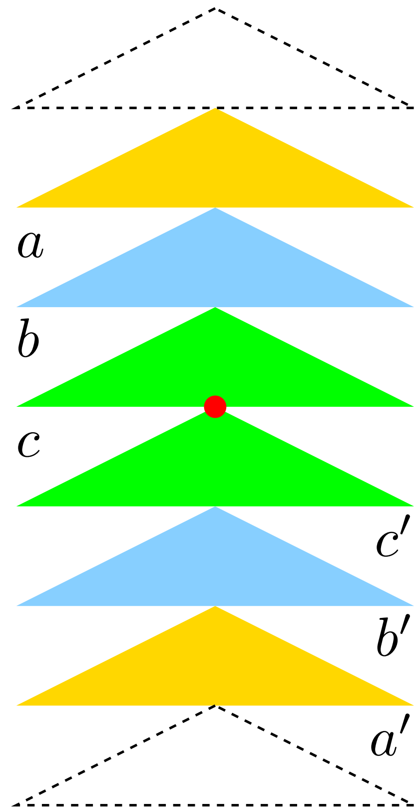


Derivation tree

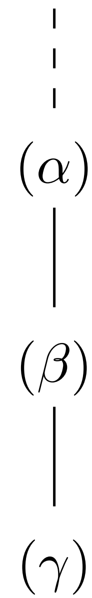
TAG derivation of copy constructions



Auxiliary trees

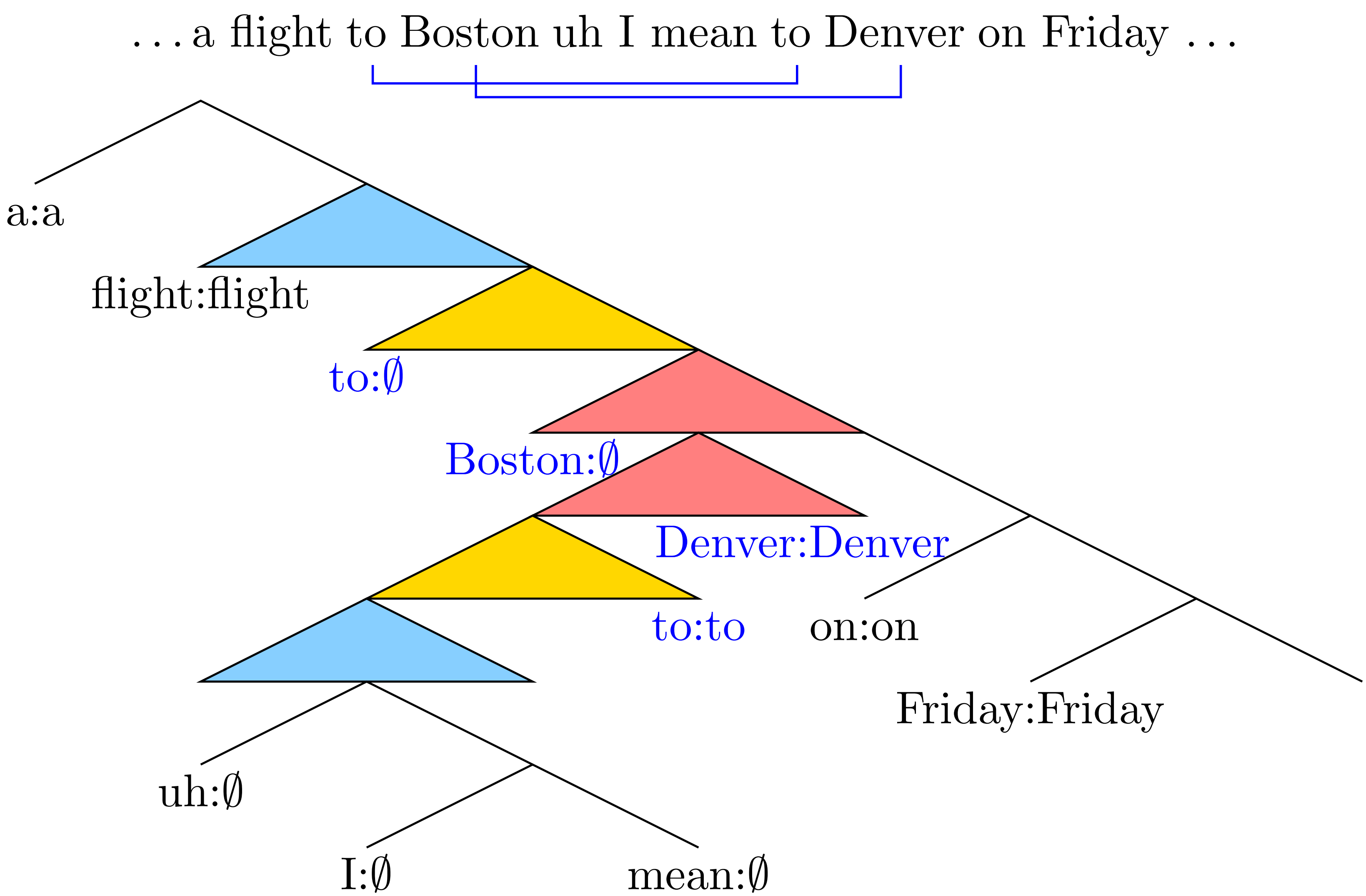


Derived tree



Derivation tree

Schematic TAG noisy channel derivation

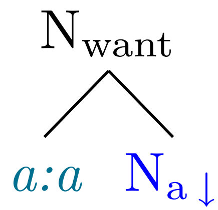


Sample TAG derivation (simplified)

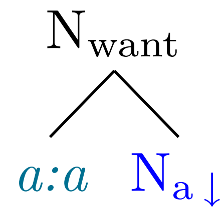
(I want) a flight to Boston uh I mean to Denver on Friday ...

Start state: $\mathrm{N}_{\mathrm{want}\downarrow}$

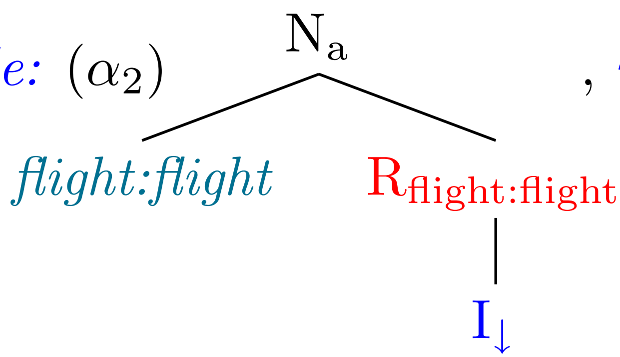
TAG rule: (α_1)



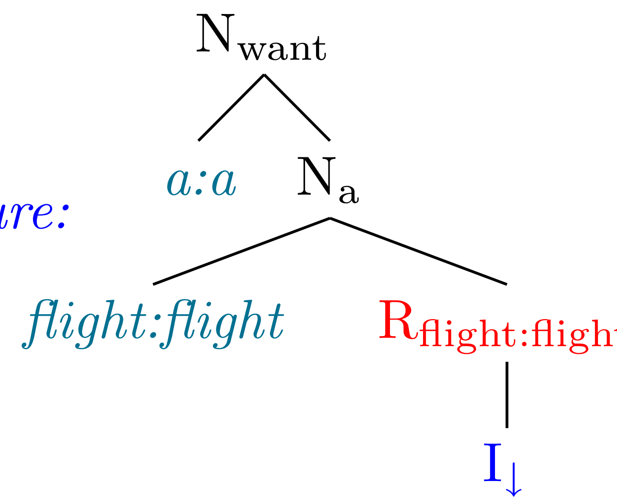
, *resulting structure:*



TAG rule: (α_2)

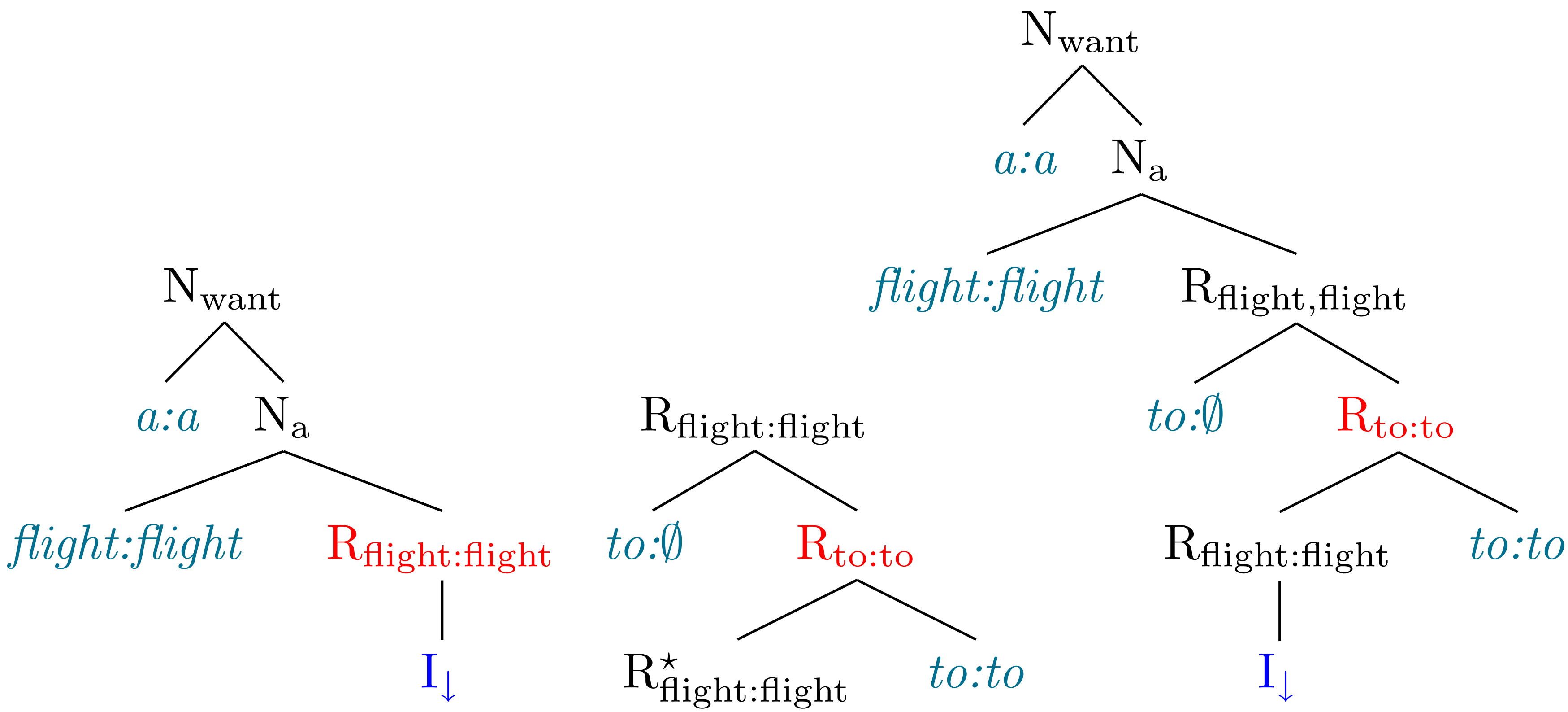


, *resulting structure:*



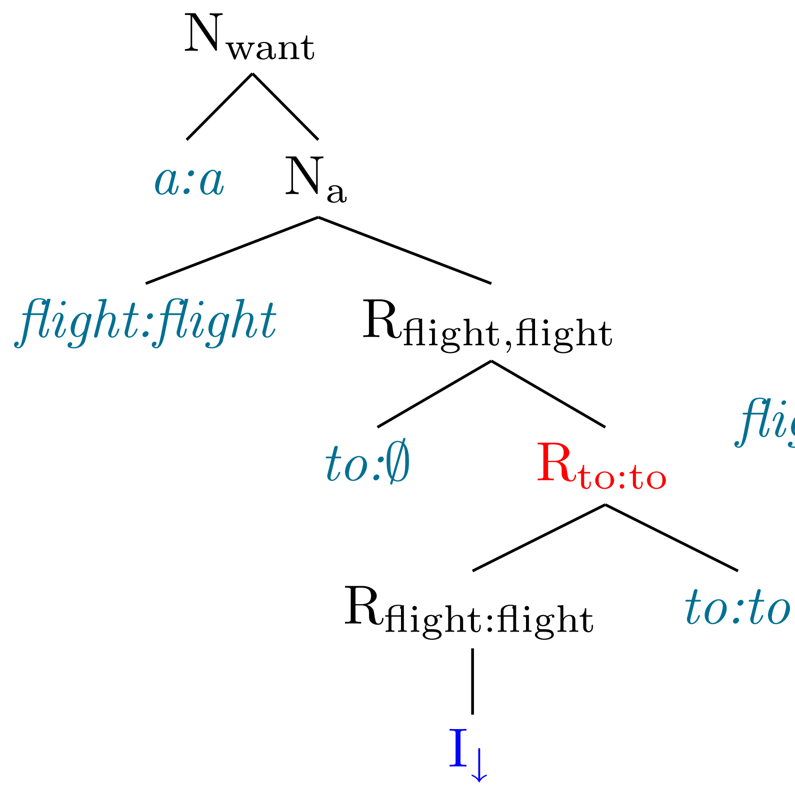
Sample TAG derivation (cont)

(I want) a flight to Boston uh I mean to Denver on Friday . . .

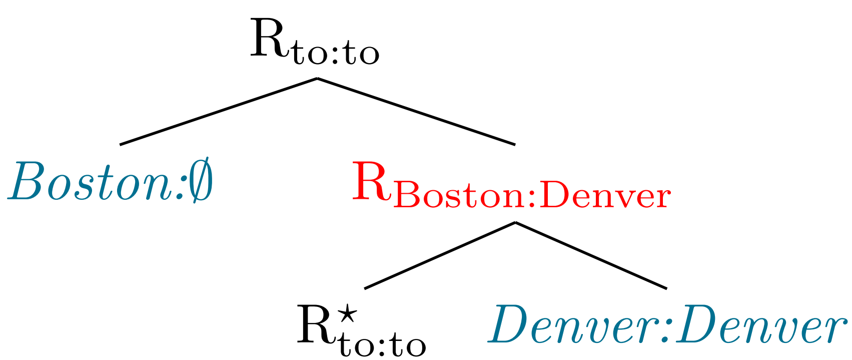


previous structure *TAG rule* (β_1) *resulting structure*

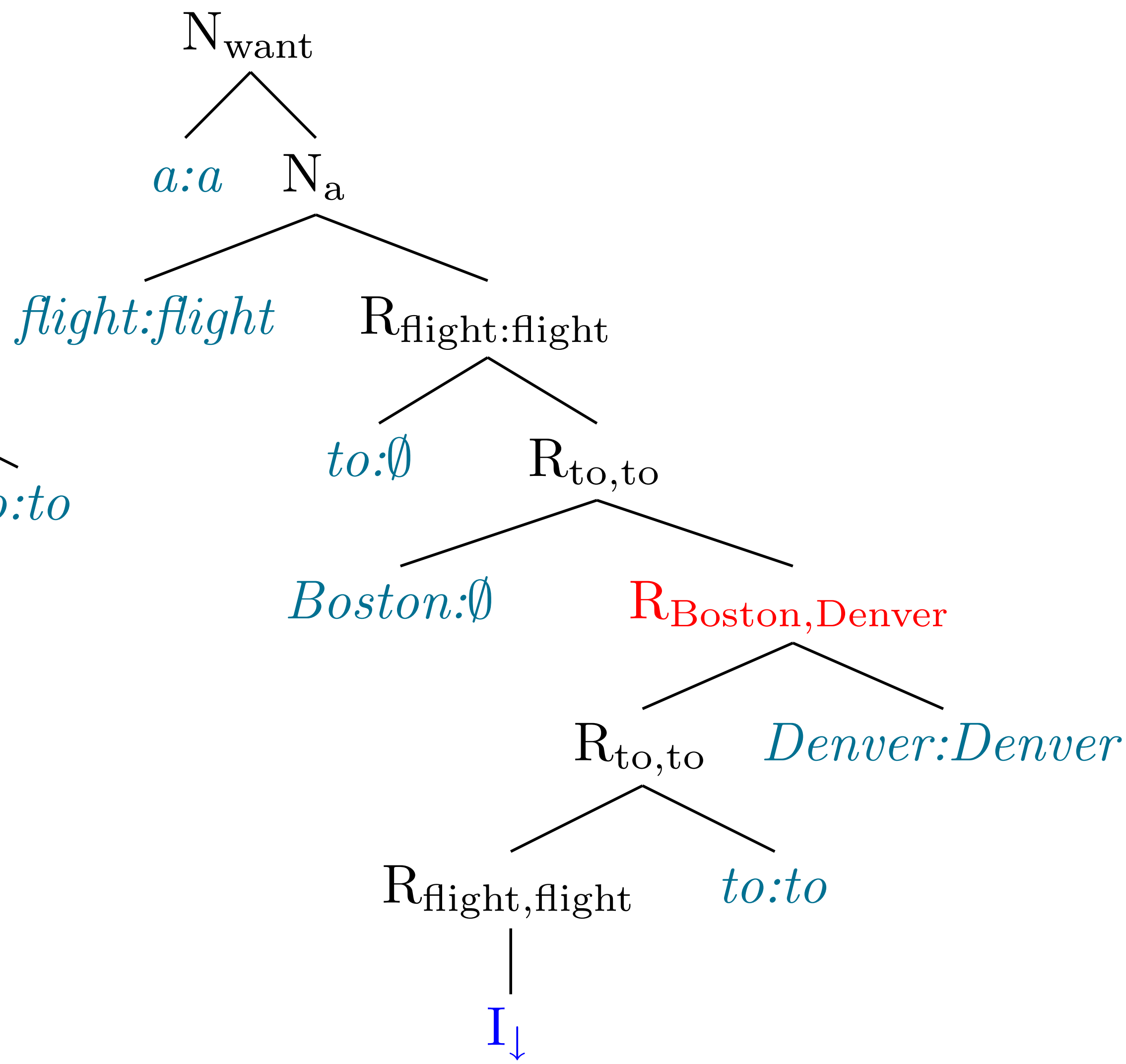
(I want) a flight to Boston uh I mean *to* Denver on Friday …



previous structure

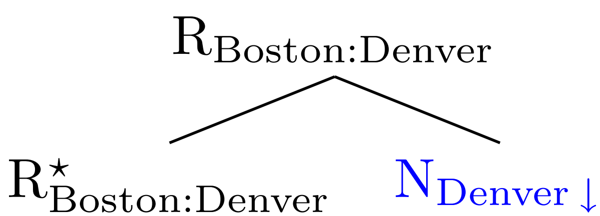


TAG rule (β_2)

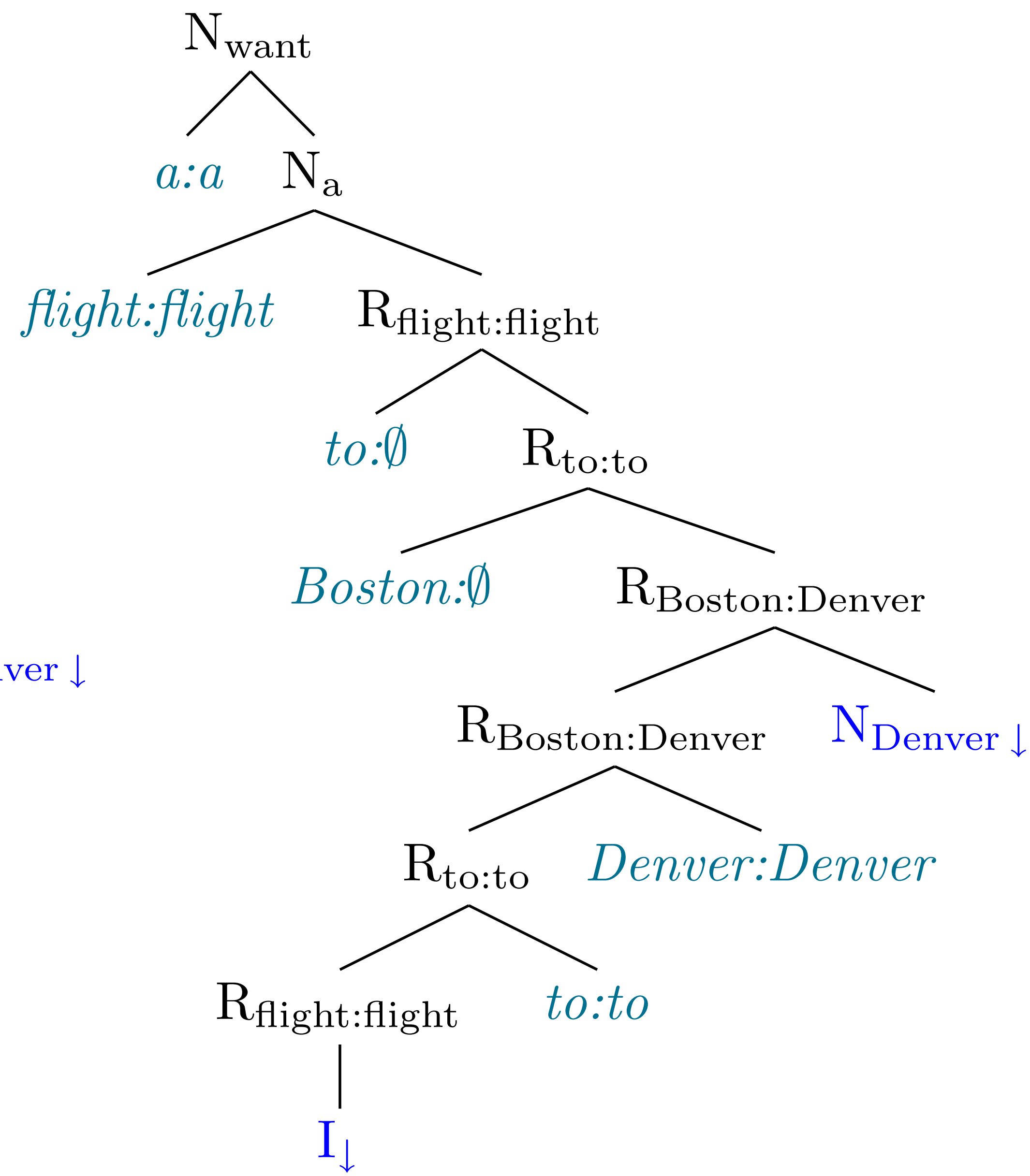


resulting structure

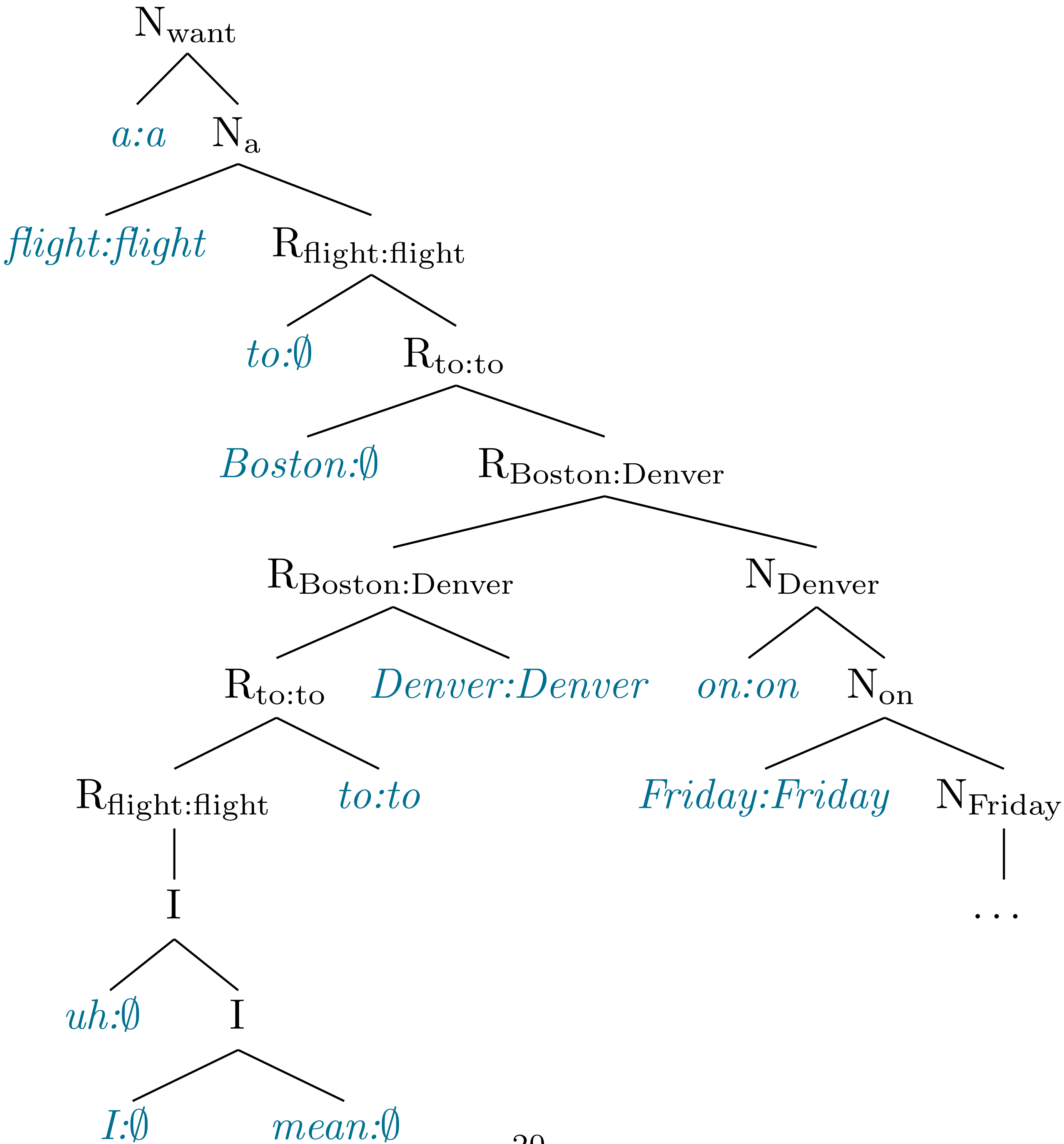
(I want) a flight to Boston uh I mean *to Denver* on Friday . . .



TAG rule (β_3)



resulting structure


N want
a:a
N a
flight:flight
R flight:flight
to:∅
R to:to
Boston:∅
R Boston:Denver
R Boston:Denver
N Denver
R to:to
Denver:Denver
on:on
N on
R flight:flight
to:to
Friday:Friday
N Friday
I
. . .
uh:∅
I
I:∅
mean:∅

Switchboard corpus data

$$\ldots \text{a flight } \underbrace{\text{to Boston,}}_{\text{Reparandum}} \; \underbrace{\text{uh, I mean,}}_{\text{Interregnum}} \; \underbrace{\text{to Denver}}_{\text{Repair}} \text{ on Friday} \ldots$$

- TAG channel model trained on the disfluency POS tagged Switchboard files sw[23]*.dps (1.3M words) which annotates reparandum, interregnum and repair

- Language model trained on the parsed Switchboard files sw[23]*.mrg with Reparandum and Interregnum removed

- 31K repairs, average repair length 1.6 words

- Number of training words: reparandum 50K (3.8%), interregnum 10K (0.8%), repair 53K (4%), overlapping repairs or otherwise unclassified 24K (1.8%)

Training data for TAG channel model

$$\ldots \text{a flight } \underbrace{\text{to Boston,}}_{\text{Reparandum}} \ \underbrace{\text{uh, I mean,}}_{\text{Interregnum}} \ \underbrace{\text{to Denver}}_{\text{Repair}} \text{ on Friday} \ldots$$

- Minimum edit distance aligner used to align reparandum and repair words
  - Prefers identity, POS identity, similar POS alignments
- Of the 57K alignments in the training data:
  - 35K (62%) are identities
  - 7K (12%) are insertions
  - 9K (16%) are deletions
  - 5.6K (10%) are substitutions
    - 2.9K (5%) are substitutions with same POS
    - 148 of the 352 substitutions (42%) in heldout data were not seen in training

Decoding using n-best rescoring

- We don't know of any efficient algorithms for decoding a TAG-based noisy channel and a parser-based language model ...
- but *the intersection of an n-gram language model and the TAG-based noisy channel is just another TAG*

⇒ Use the parser language model to rescore the 20-best bigram language model results:

  – Use the *bigram language model* with a *dynamic programming search* to find the 20 best analyses of each string

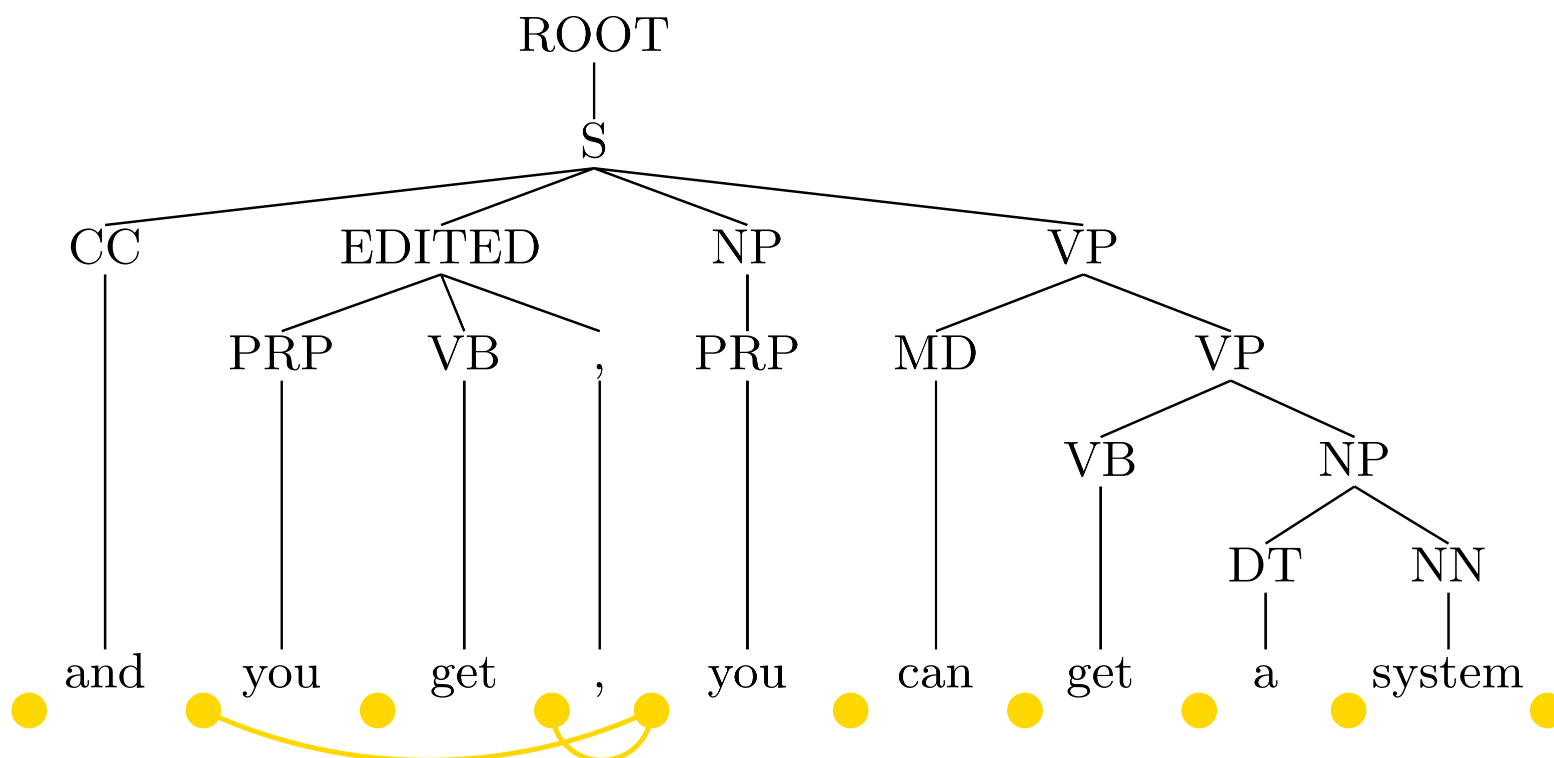
  – Parse each of these using the parser-based language model

  – Select the overall highest-scoring analysis using the parser probabilities and the TAG-based noisy channel scores

See: Collins (2000) "Discriminative Reranking for Natural Language Parsing", Collins and Koo (to appear) "Discriminative Reranking for Natural Language Parsing"

Modified labeled precision/recall evaluation

- Goal: Don't penalize misattachment of EDITED nodes
- String positions on either side of EDITED nodes *in the gold-standard corpus tree* are equivalent (just like punctuation in PARSEVAL)



Charniak and Johnson (2001) "Edit detection and parsing for transcribed speech"

Empirical results

- Training and testing data has *partial words and punctuation removed*
- CJ01′ is the Charniak and Johnson 2001 word-by-word classifier trained on new training and testing data
- Bigram is the Viterbi analysis using dynamic programming decoding with bigram language model
- Trigram and Parser are results of 20-best reranking using trigram and parser language models

	CJ01′	Bigram	Trigram	Parser
Precision	0.951	0.776	0.774	0.820
Recall	0.631	0.736	0.763	0.778
F-score	0.759	0.756	0.768	0.797

Conclusion and future work

- It is possible to detect and excise speech repairs with reasonable accuracy
- We can incorporate the very different syntactic and repair structures in a single *noisy channel model*
- Using a better language model improves overall performance
- It might be interesting to make the channel model *sensitive to syntactic structure* to capture the relationship between syntactic context and the location of repairs
- A *log-linear model* should permit us to integrate a wide variety of interacting syntactic and repair features
- *There are lots of interesting ways of combining speech and parsing!*

Estimating the model from data

$$\ldots\text{a flight } \underbrace{\text{to Boston,}}_{\text{Reparandum}}\ \underbrace{\text{uh, I mean,}}_{\text{Interregnum}}\ \underbrace{\text{to Denver}}_{\text{Repair}} \text{ on Friday} \ldots$$

$\mathrm{P}_n(\textsf{repair}|\textit{flight})$ The probability of a repair beginning after *flight*

$\mathrm{P}(m|\textit{Boston}, \textit{Denver})$, where $m \in \{\textsf{copy}, \textsf{substitute}, \textsf{insert}, \textsf{delete}, \textsf{nonrepair}\}$:
The probability of repair type m when the last reparandum word was *Boston* and the last repair word was *Denver*

$\mathrm{P}_w(\textit{tomorrow}|\textit{Boston}, \textit{Denver})$ The probability that the next reparandum word is *tomorrow* when the last reparandum word was *Boston* and last repair word was *Denver*

The TAG rules and their probabilities

$$P\left(\begin{array}{ccc} & \mathrm{N_{want}} & \\ \diagup & & \diagdown \\ \textit{a:a} & & \mathrm{N_{a}}\downarrow \end{array}\right) = (1 - P_n(\mathsf{repair}|a))$$

$$P\left(\begin{array}{ccc} & \mathrm{N_{a}} & \\ \diagup & & \diagdown \\ \textit{flight:flight} & & \mathrm{R_{flight:flight}} \\ & & | \\ & & \mathrm{I}_{\downarrow} \end{array}\right) = P_n(\mathsf{repair}|\textit{flight})$$

- These rules are just the TAG formulation of a HMM.

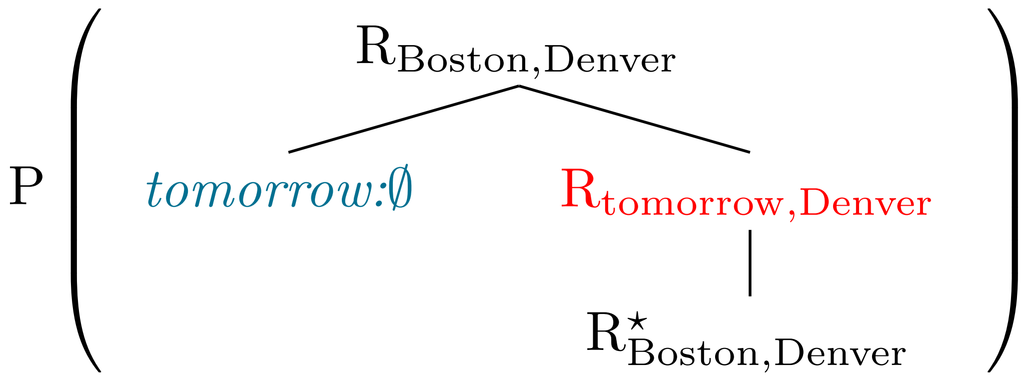
The TAG rules and their probabilities (cont.)

$$\mathrm{P}\left(\begin{array}{cccc} & \mathrm{R}_{\text{flight:flight}} & & \\ \textit{to:}\emptyset & & \mathrm{R}_{\text{to:to}} & \\ & \mathrm{R}^{\star}_{\text{flight:flight}} & & \textit{to:to} \end{array}\right) = \mathrm{P}_r(\mathsf{copy}|\textit{flight}, \textit{flight})$$

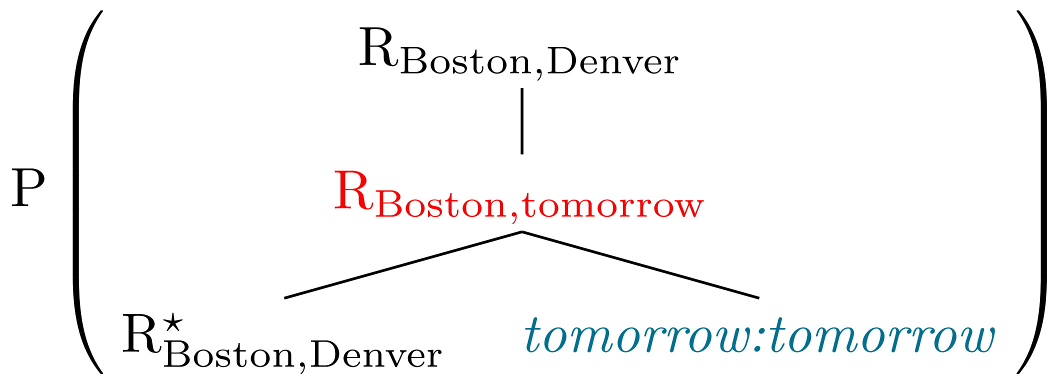
$$\mathrm{P}\left(\begin{array}{cccc} & \mathrm{R}_{\text{to:to}} & & \\ \textit{Boston:}\emptyset & & \mathrm{R}_{\text{Boston:Denver}} & \\ & \mathrm{R}^{\star}_{\text{to:to}} & & \textit{Denver:Denver} \end{array}\right) = \begin{array}{l} \mathrm{P}_r(\mathsf{substitute}|\textit{to}, \textit{to}) \\ \mathrm{P}_w(\textit{Boston}|\textit{to}, \textit{to}) \end{array}$$

- Copies generally have higher probability than substitutions

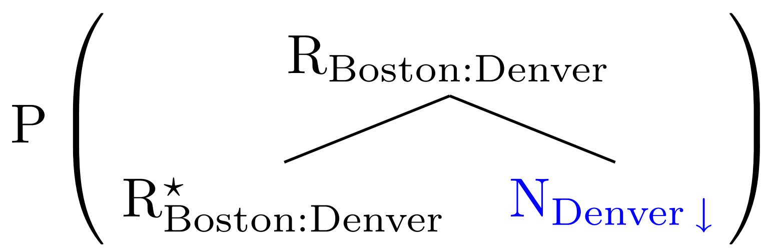
The TAG rules and their probabilities (cont.)



$$= \mathrm{P}_r(\mathsf{insert}|Boston, Denver)\ \mathrm{P}_w(tomorrow|Boston, Denver)$$



$$= \mathrm{P}_r(\mathsf{delete}|Boston, Denver)$$



$$= \mathrm{P}_r(\mathsf{nonrepair}|Boston, Denver)$$

Decoding with a bigram language model

- We could search for the most likely parses of each sentence ...
- or alternatively *interpret the dynamic programming table directly*:
  1. compute the probability that each triple of adjacent substrings can be analysed as a reparandum/interregnum/repair
  2. divide by the probability that the substrings do not contain a repair
  3. if these *odds* are greater than a fixed threshold, identify this reparandum as EDITED.
  4. find most highly scoring combination of repairs
- Advantages of the more complex approach:
  - Doesn't require parsing the whole sentence (rather, only look for repairs up to some maximum size)
  - Adjusting the odds threshold trades precision for recall
  - Handles *overlapping repairs* (where the repair is itself repaired)

  [[What did + what does he] + what does she] want?

(Standard) labeled precision/recall

- *Precision* = # correct nodes/# nodes in parse trees
- *Recall* = # correct nodes/# nodes in corpus trees
- A parse node p is correct iff there is a node c in the corpus tree such that
  - $label(p) \equiv label(c)$ (where ADVP $\equiv$ PRT)
  - $left(p) \equiv_r left(c)$ and $right(p) \equiv_r right(c)$
- $\equiv_r$ is an equivalence relation on string positions

I like , but Sandy hates , beans