Natural Language Dependency Parsing

SPFLODD

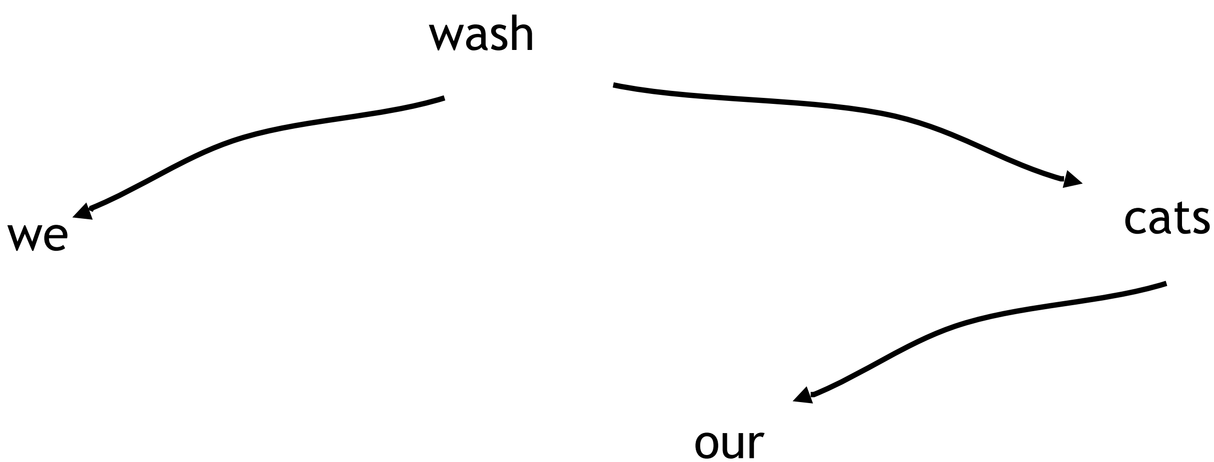
October 6, 2015

Dependency Grammar

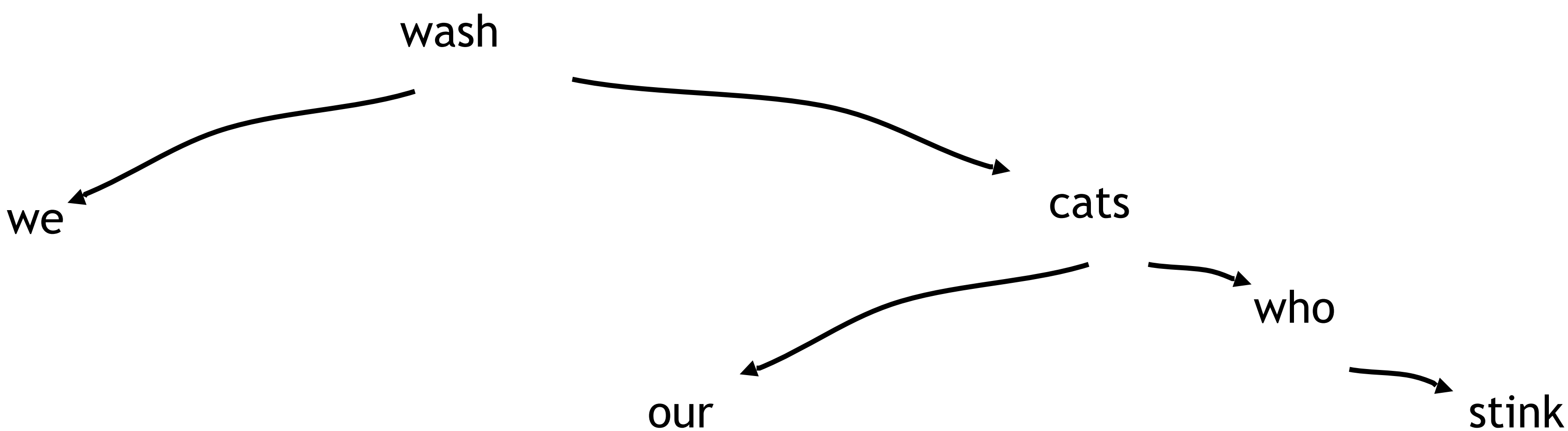
- A variety of theories and formalisms
- Focus on relationship between **words** and their syntactic relationships
- Correlates with study of languages that have free(r) word order (e.g., Czech)
- Lexicalization is central, phrases secondary

- We will talk about **bare bones** dependency trees (Eisner, 1996), then consider adding dependency labels

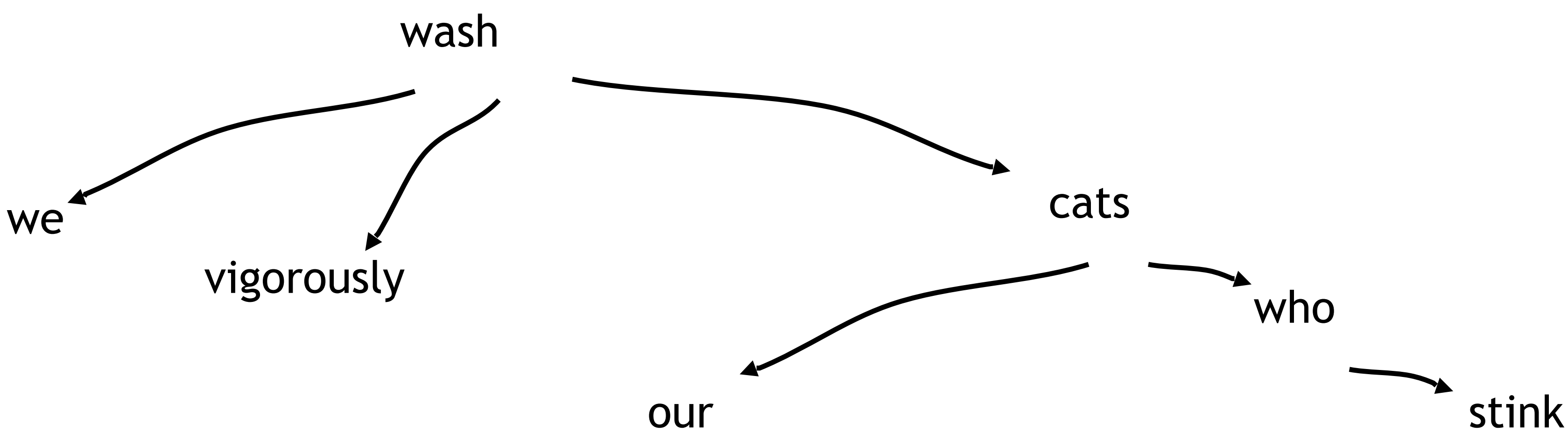
Bare Bones Dependency Parse



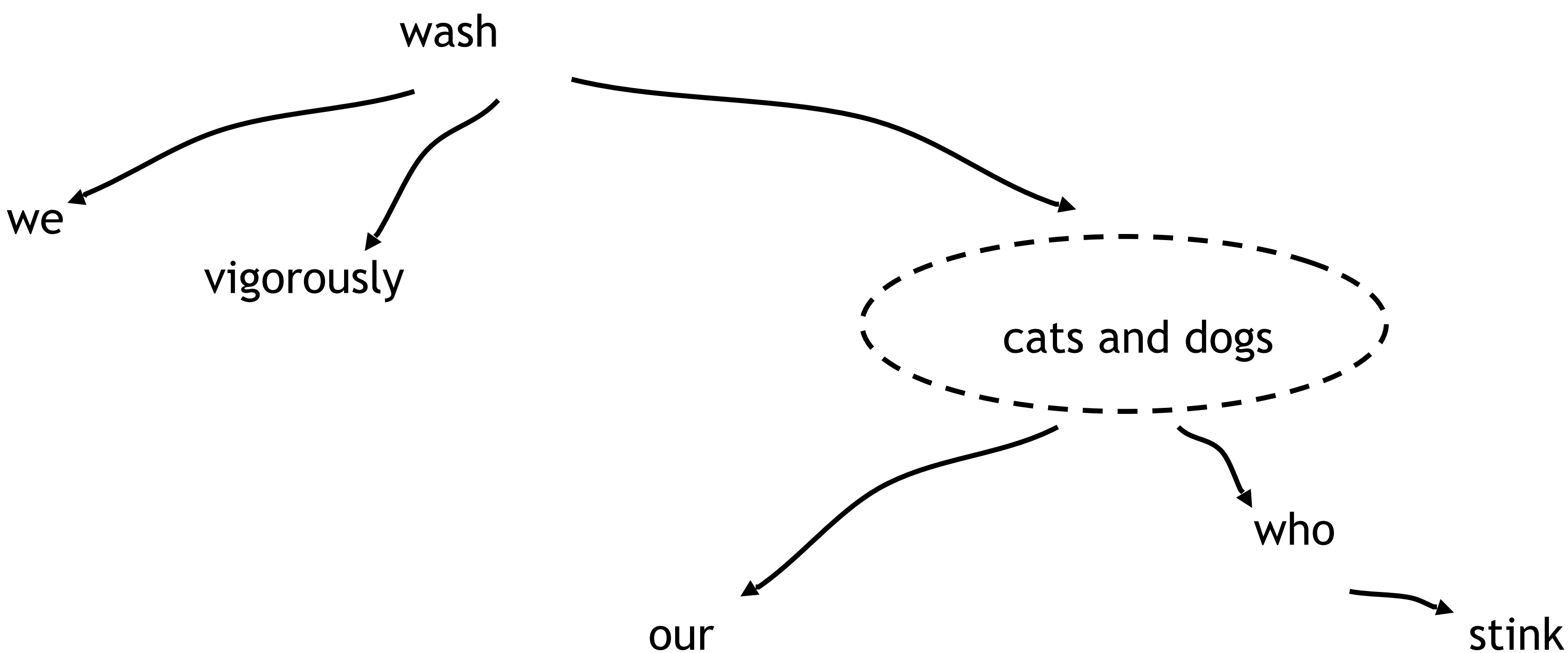
Bare Bones Dependency Parse



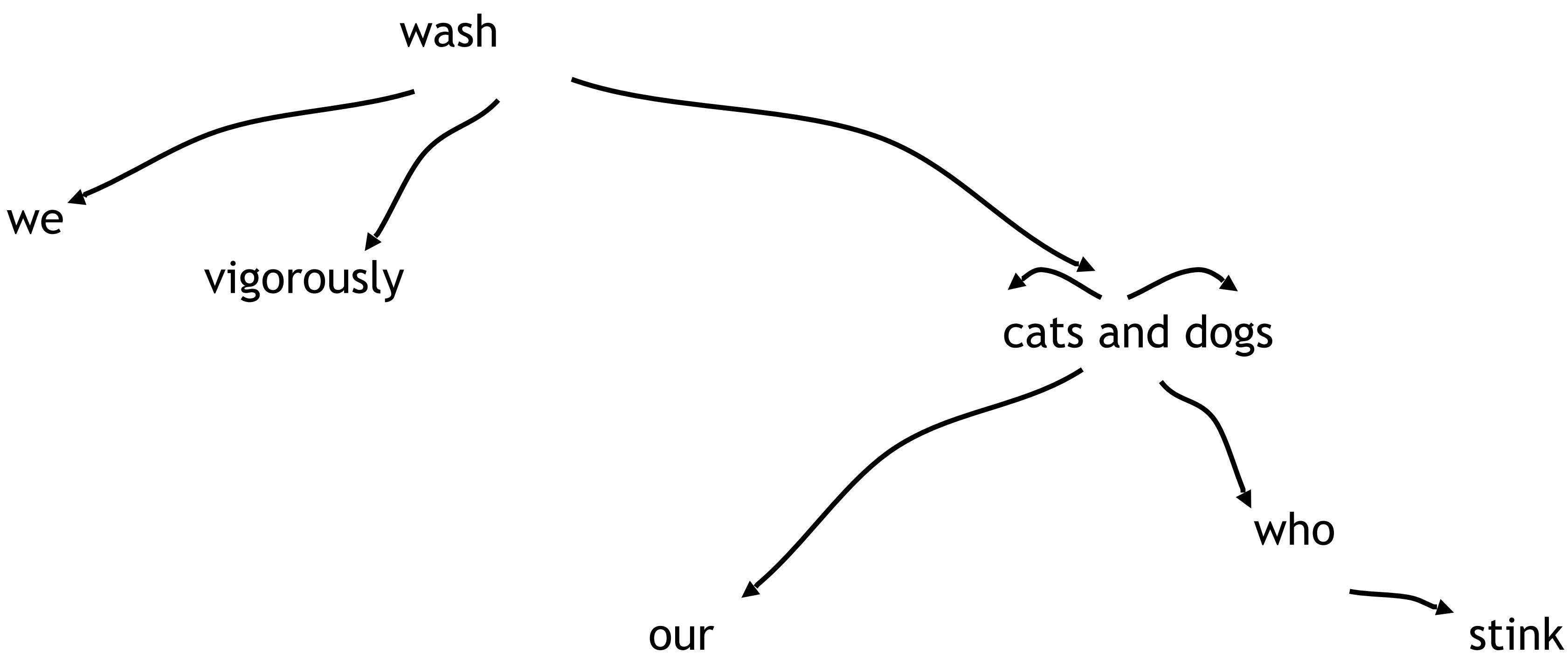
Bare Bones Dependency Parse



Bare Bones Dependency Parse



Bare Bones Dependency Parse



Bare Bones Dependencies and Labels

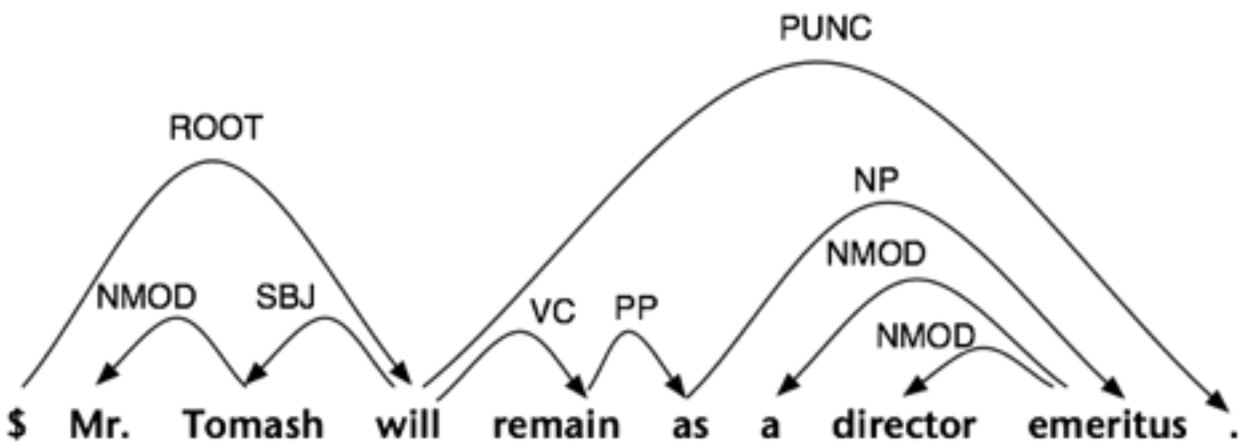
- The way to represent a lot of phenomena is clear (predicate-argument and modifier relationships)
- Conjunctions pose a problem
- Sometimes words that “should” be connected are not, because of the single-parent rule
- From bare bones to **labels**:
  - consider labeled edges
  - most algorithms can be easily extended for labeled dependency parsing
- Linguistically imperfect, but computationally attractive

Evaluation

- Attachment accuracy
  - Labeled
  - Unlabeled

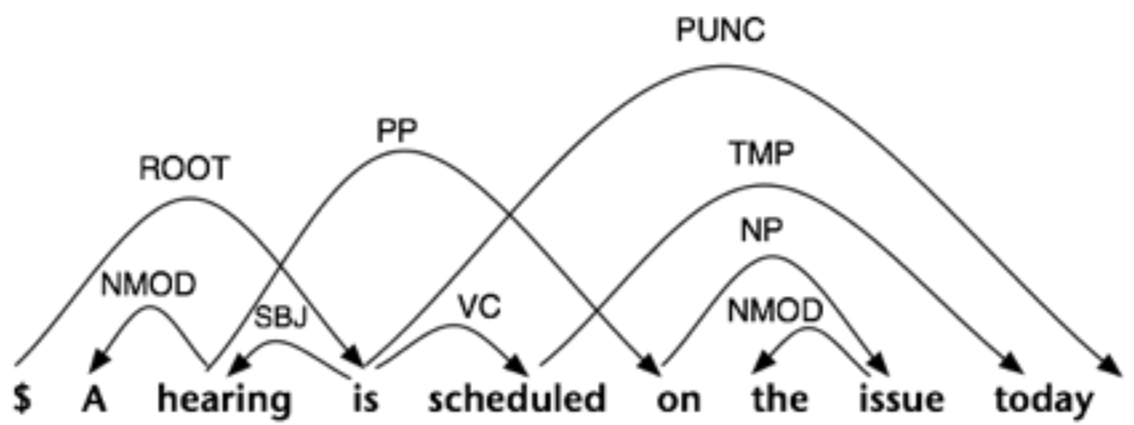
Dependencies and Context-Freeness

- **Projective** dependency trees are ones where edges don't cross
- Projective dependency parsing means searching only for projective trees
- English is mostly projective...



Dependencies and Context-Freeness

- But not entirely!



- Dependencies constructed through simple means from the Penn treebank will be projective. Why?

Dependencies and Context-Freeness

- Other languages are arguably less projective

- **Projective** dependency grammars generate **context-free** languages
- **Non-projective** dependency grammars can generate **context-sensitive** languages

Projective Dependency Parsing

- Major assumption: edge-factored model

$$p(\tau, w_1^n) \propto \prod_{i=1}^{n} \phi(w_1^n, \tau(w_i))$$

- Carroll and Charniak (1992) described a PCFG that has this property
- Eisner (1996) described several stochastic models for generating projective trees like this
- You should see that this is a linear model with a certain kind of feature locality
- We're not going to go into the details of the features that have been proposed!

Graph-based vs. Transition-based

- All models above optimize a global score and resort to local features
  - These are sometimes known as **graph-based models.**
- Just like in the phrase-structure/constituent world, there are also approaches that use shift-reduce algorithms.
- With good statistical learning methods, you can get very high performance using **greedy** search without back-tracking!
- "Local decisions, global features"
  - These are known as **transition-based models**; they reduce a structured problem to a lot of classification decisions, kind of like MEMMs.

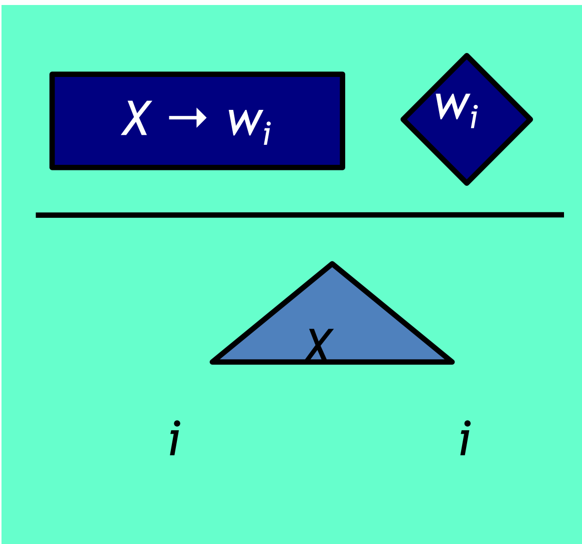
Algorithms != Models

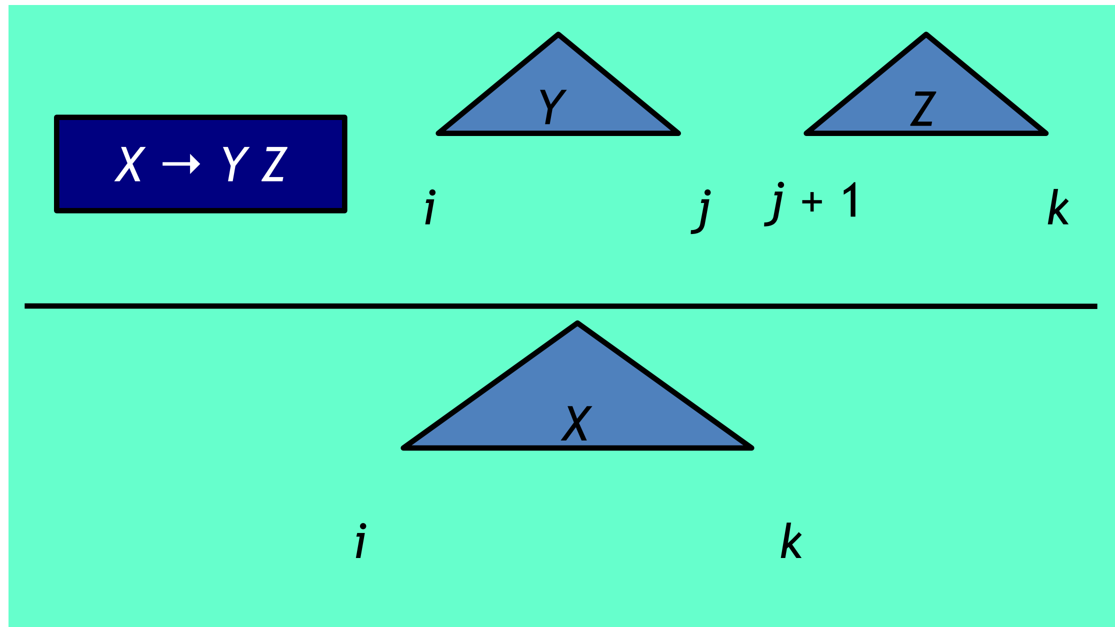
- As in HMMs, PCFGs, etc., the algorithms we need depend on the independence assumptions, **not** on the specific formulation of the statistical scores.
- We assume, from here on, that the scores are factored by dependency tree edges.

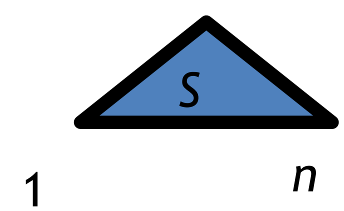
$$s(\tau) = \sum_{i=1}^{n} s(w_1^n, \tau(w_i))$$

- Projective algorithm (Eisner, 1996)
- Non-projective algorithm (McDonald et al., 2005)

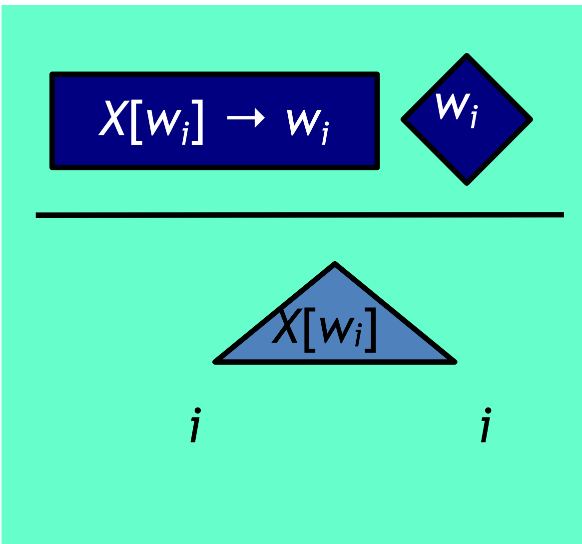
CKY

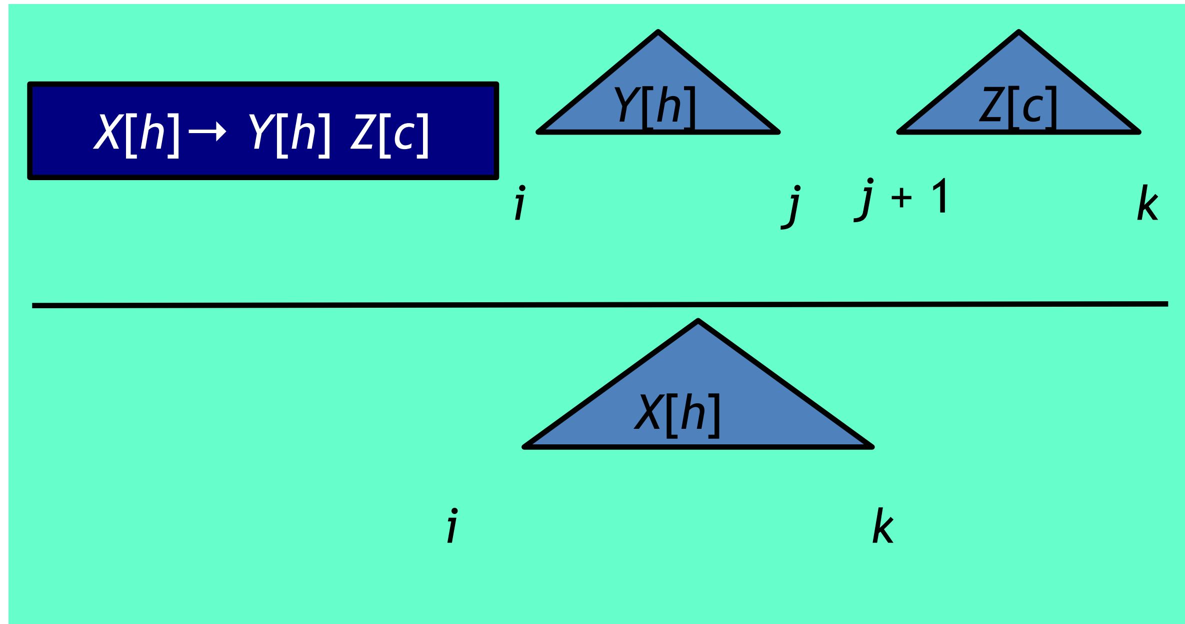


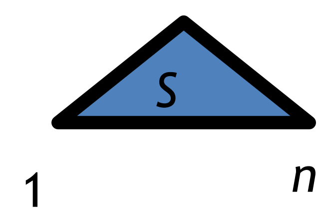




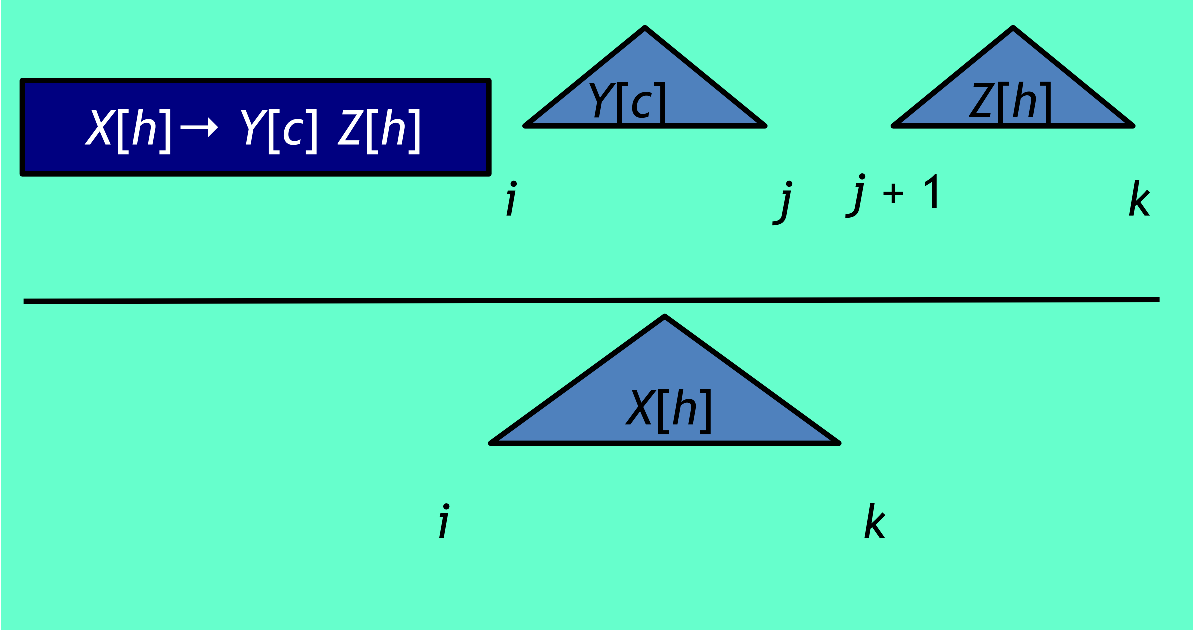
CKY with Heads



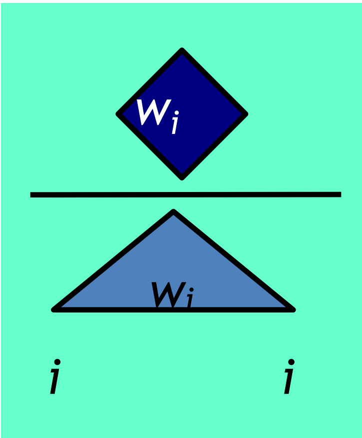


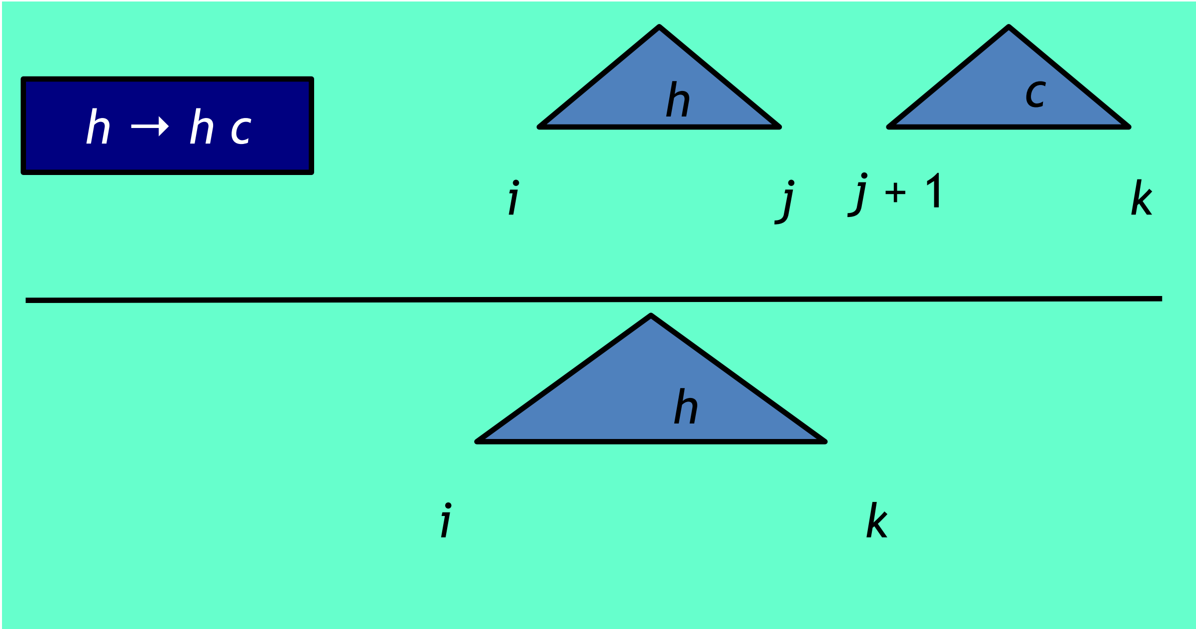


CKY with Heads (one more rule)



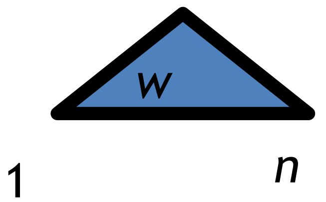
CKY with Heads, without Nonterminals





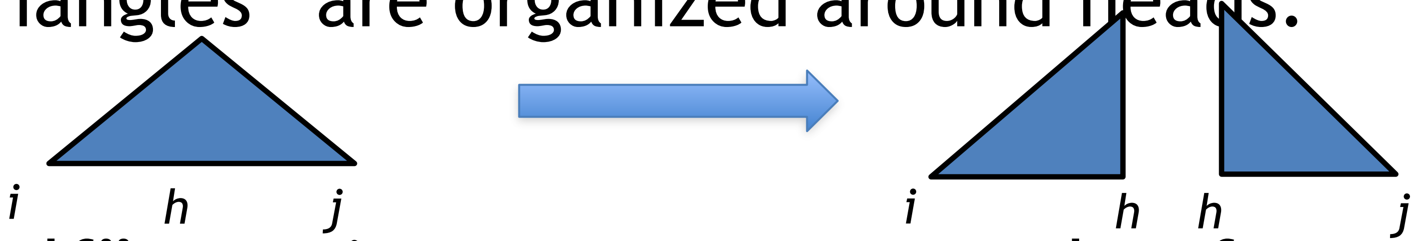
*Plus the rule for $h \rightarrow c\ h$.

What's the runtime?

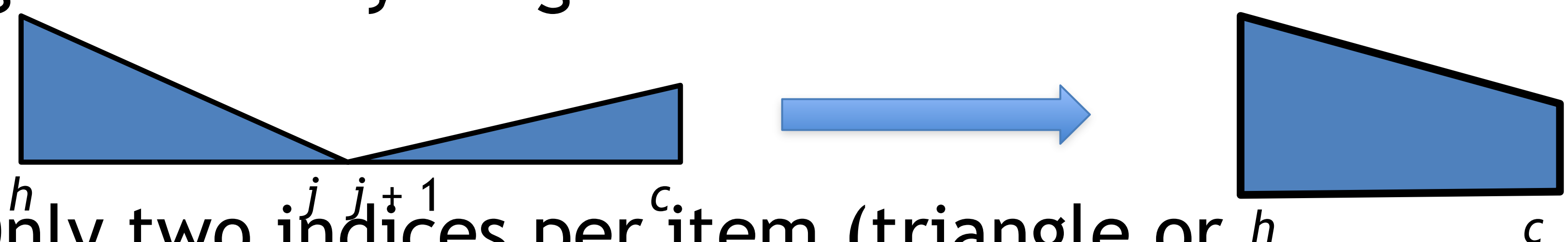


Eisner's (1996) Algorithm

- Restructure the computation so that "triangles" are organized around heads.

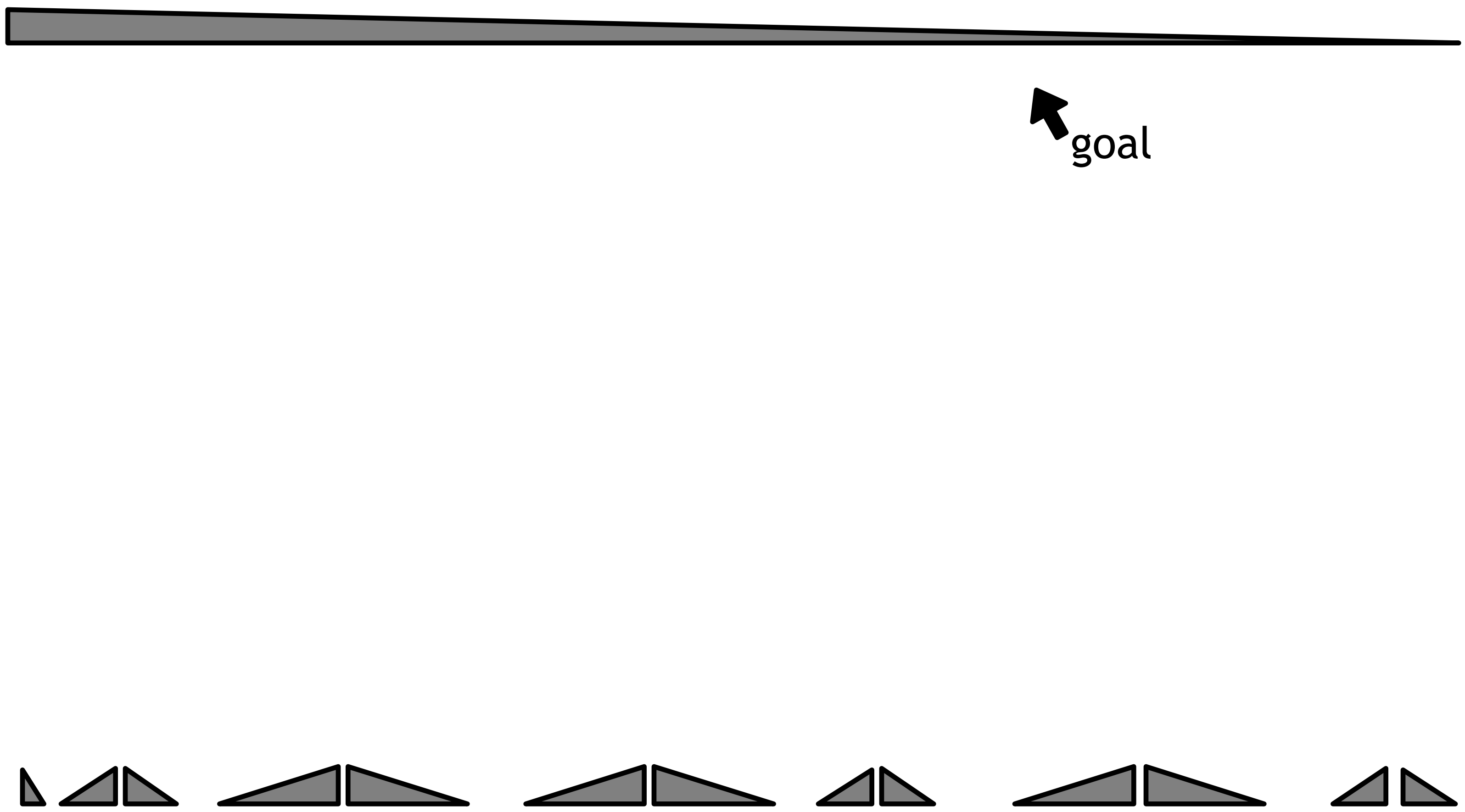


- "Half" constituents get put together from head outward; attaching a child to a parent ignores everything not between the two.



- Only two indices per item (triangle or trapezoid).

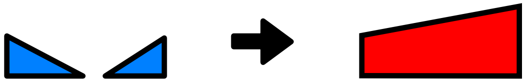
Example of the Eisner Algorithm

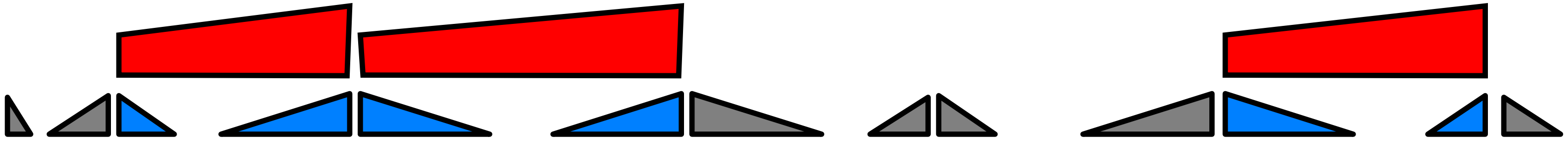


$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

Attach:



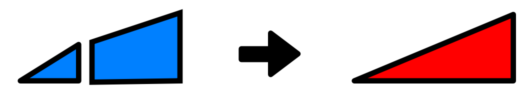


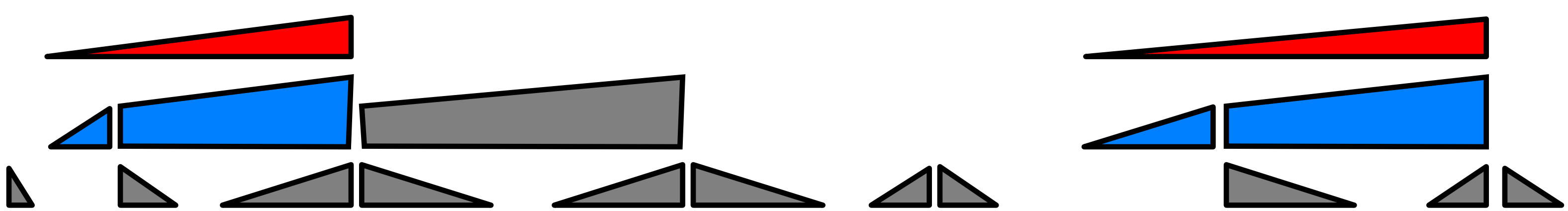
$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

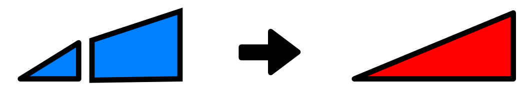
Complete: 

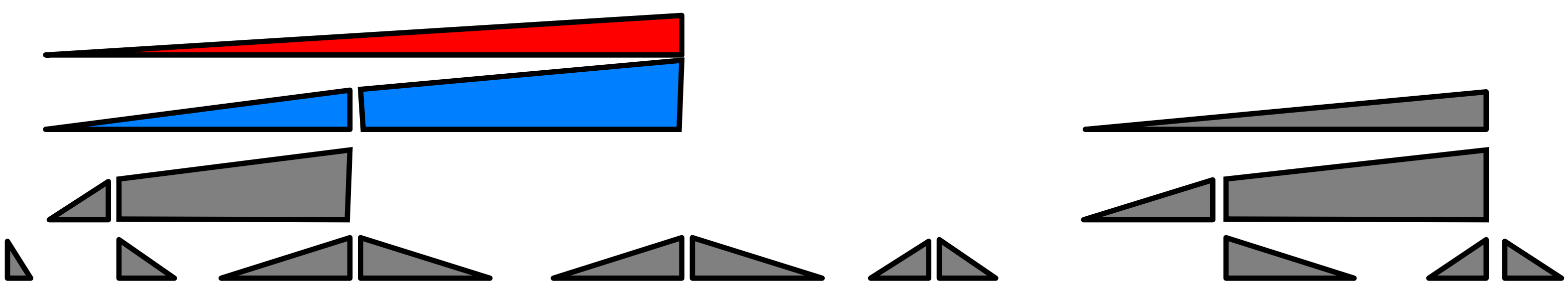


$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

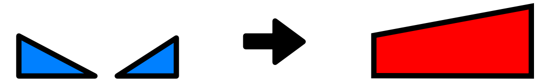
Complete:

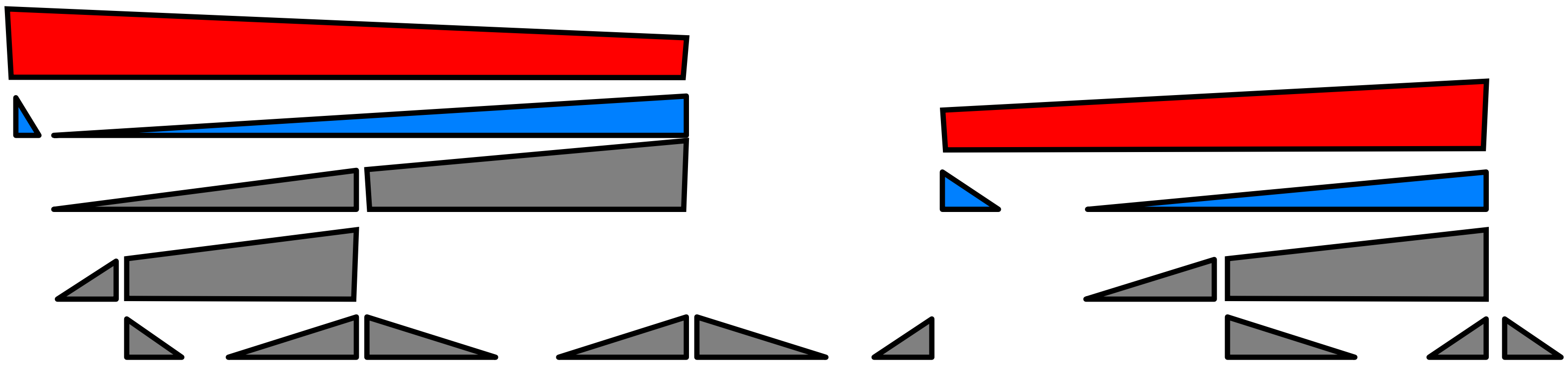




$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

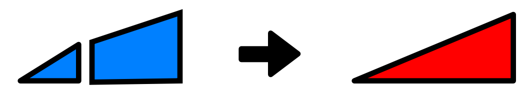
Attach: 

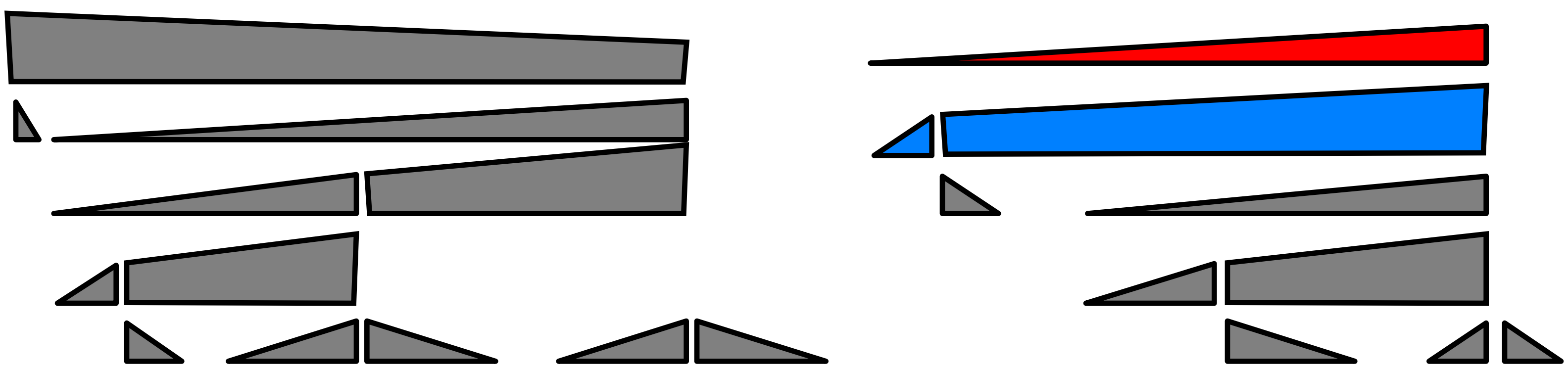


$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

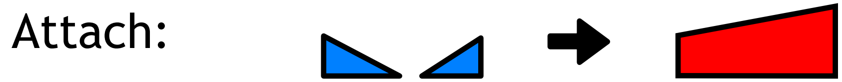
Complete:

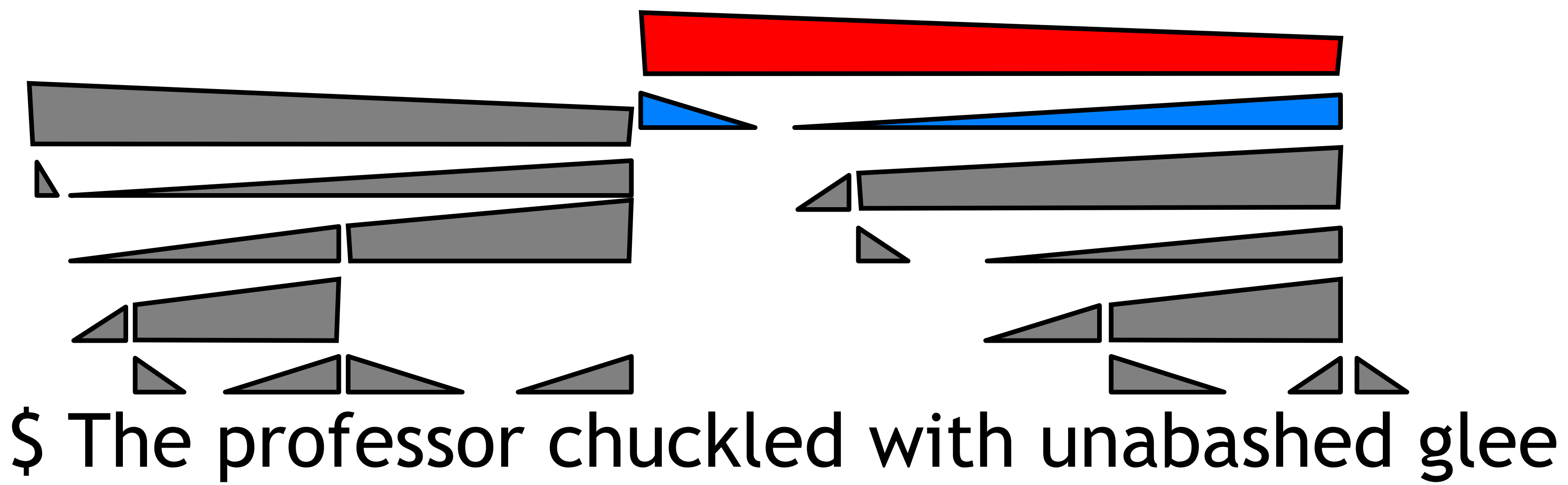




$ The professor chuckled with unabashed glee

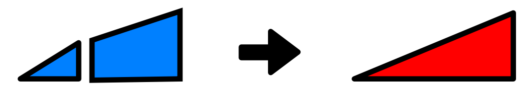
Example of the Eisner Algorithm

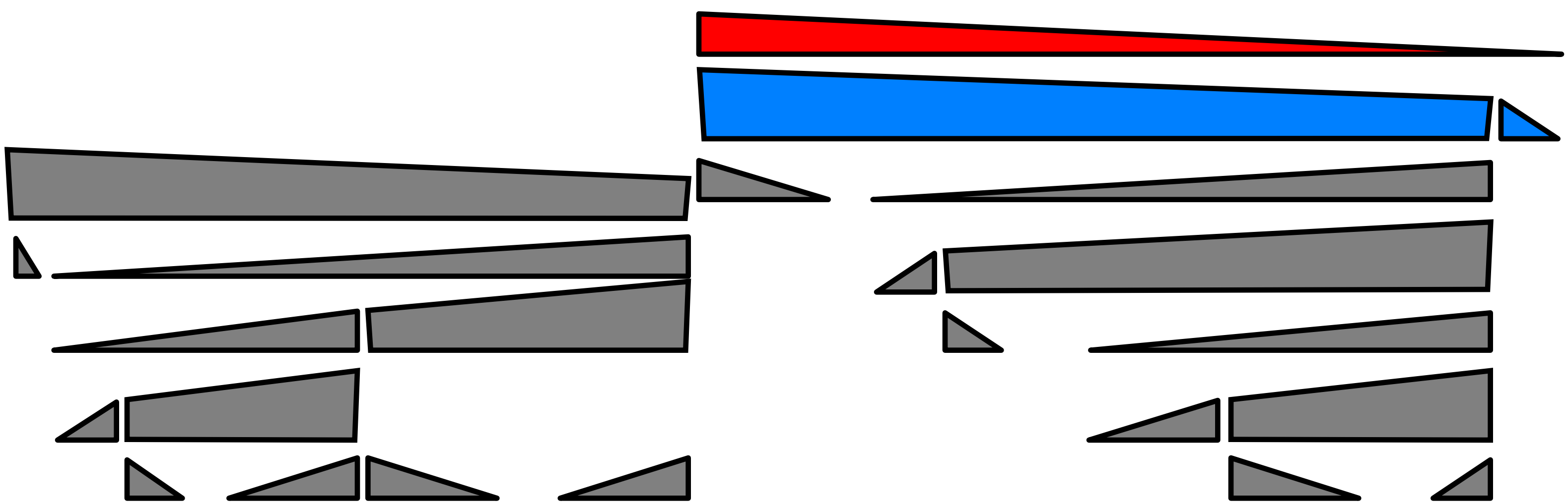




Example of the Eisner Algorithm

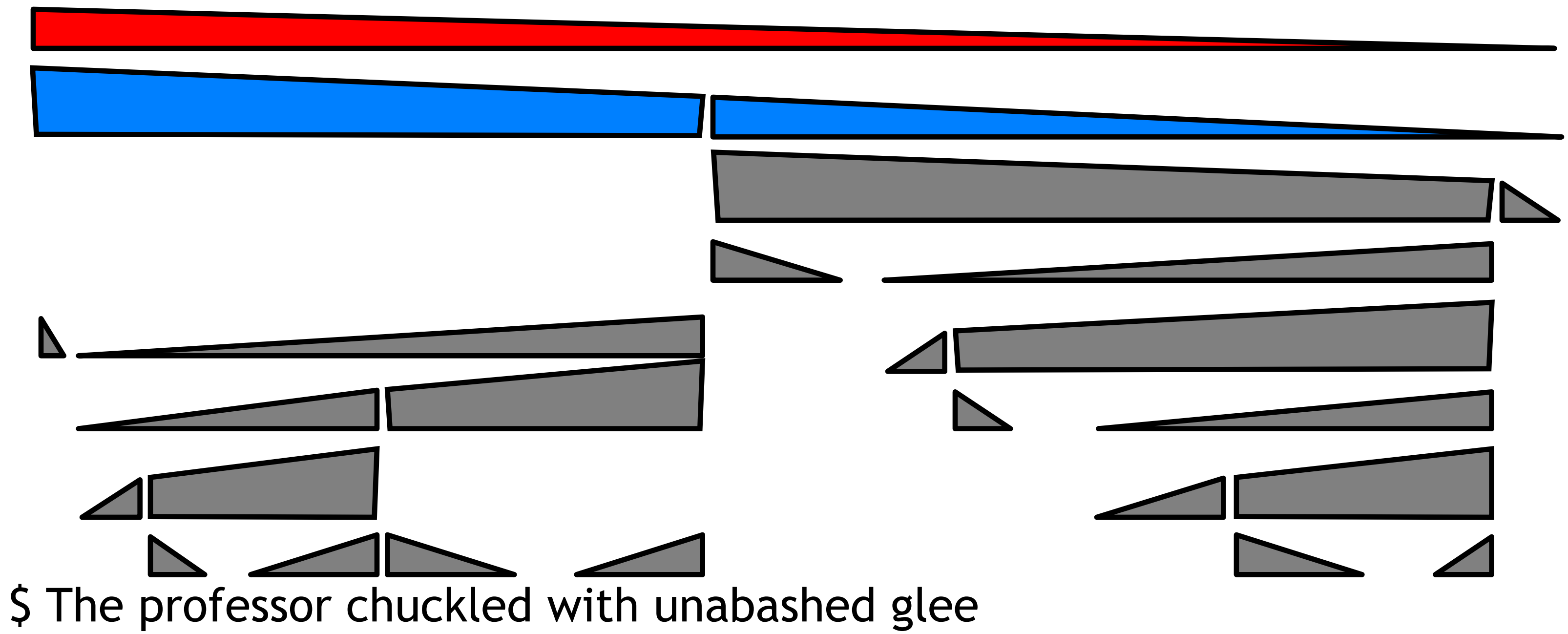
Complete:



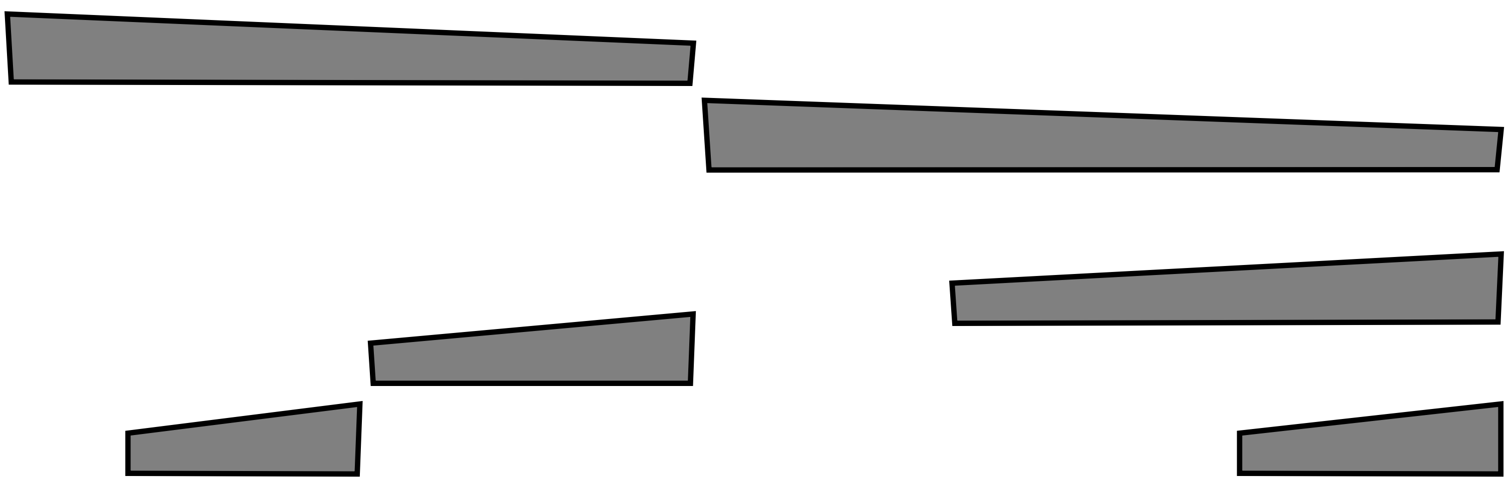


$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm

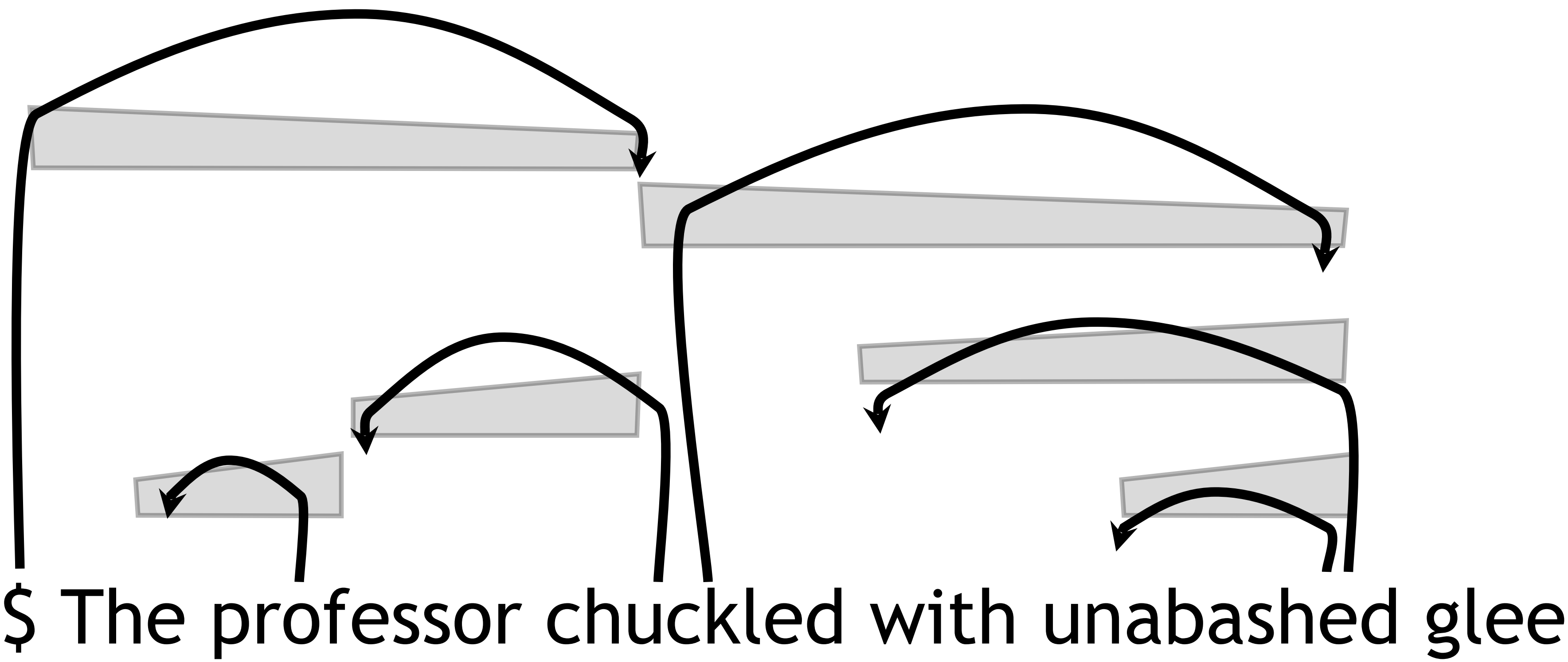


Example of the Eisner Algorithm



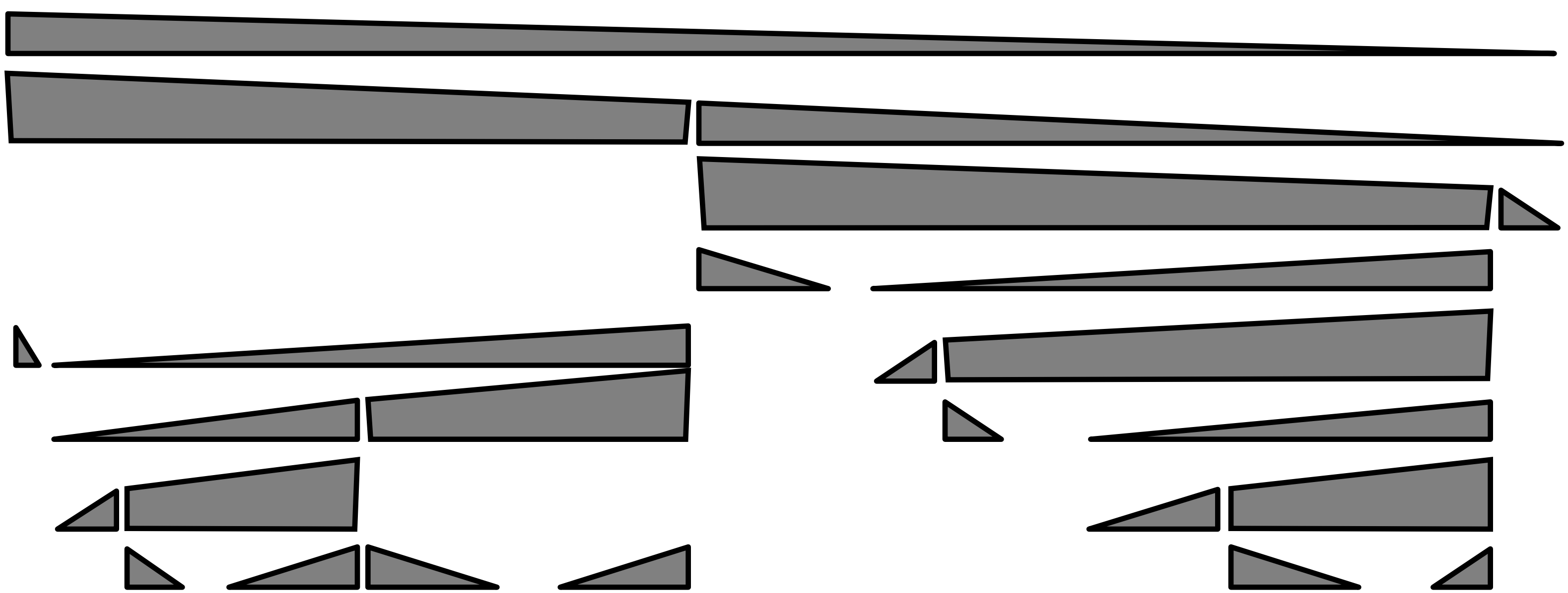
$ The professor chuckled with unabashed glee

Example of the Eisner Algorithm



Example of the Eisner Algorithm

Of course, we have to build a lot of other triangles and trapezoids if we want to be sure we have the *best* parse.



$ The professor chuckled with unabashed glee

Punchline

- Rethinking the algorithm in terms of attachments rather than constituents gives us an asymptotic savings!
- Bare bones, projective dependency parsing is $O(n^3)$

- What about non-projectivity?

Non-projective Dependency Parsing (McDonald et al., 2005)

- Key idea: a non-projective dependency parse is a **directed spanning tree** where
  - vertices = words
  - directed edges = parent-to-child relations
- Well-known problem: minimum-cost spanning tree
- Solution: Chu-Liu-Edmonds algorithm (cubic)
  - Tarjan: quadratic for dense graphs (like ours)
- Good news: fast! can now recover non-projective trees!
- Bad news: much larger search space, potential for error

Breaking Independence Assumptions

- Adding labels doesn't fundamentally change Eisner or MST
- What about edge-factoring?
- Projective case: local statistical dependence among same-side children of a given head - still cubic (Eisner and Satta, 1999).
- Non-projective parsing with any kind of second-order features (e.g., on adjacent edges) is NP-hard.
  – McDonald explored approximations in his thesis
  – Find the best projective parse and then rearrange the edges as long as the score improves - $O(n^3)$
  – ILP (Martins et al., 2009)

CoNLL 2006 & 2007

- 2006 and 2007: dependency parsing on a variety of languages was the shared task at CoNLL - a few dozen systems.
- Many of the datasets are freely available.
- Parsing papers now typically evaluate on most or all of these datasets.