The Real Story on Synchronous Rewriting Systems

Daniel Gildea

Computer Science Department

University of Rochester

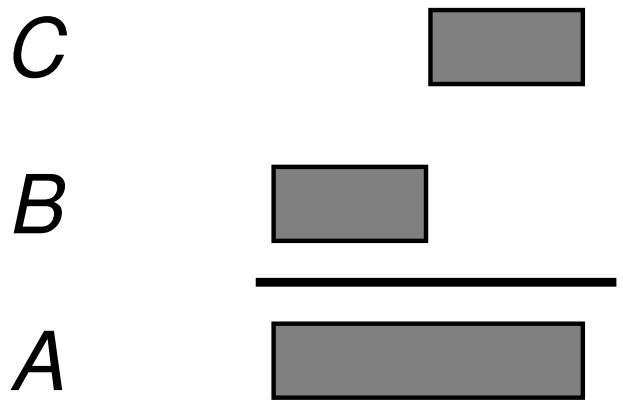
Overview

- Factorization
- Tree Decomposition
- Factorization of LCFRS

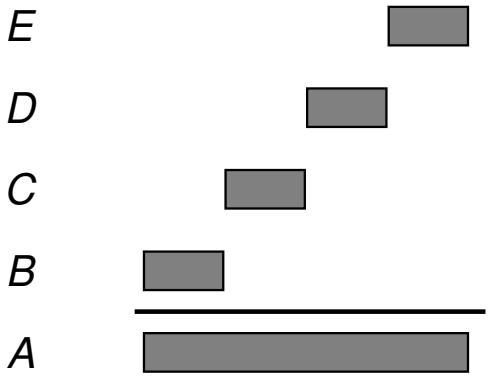
CYK Parsing

$A \rightarrow B\,C$

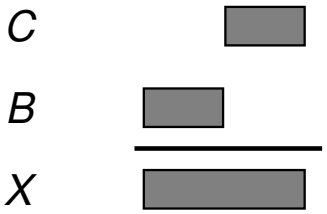
$$\frac{w_2\colon\ [C, x_1, x_2] \qquad w_1\colon\ [B, x_0, x_1]}{w_0 w_1 w_2\colon\ [A, x_0, x_2]}$$



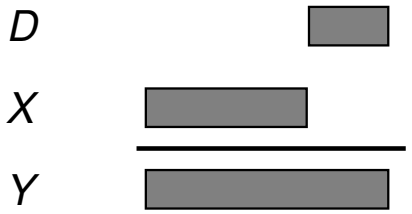
Binarization of CFG



$A \rightarrow B\ C\ D\ E$



$X \rightarrow B\ C$



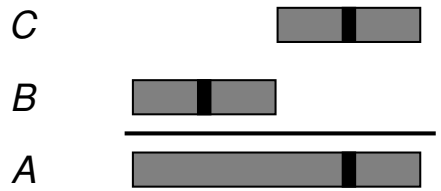
$Y \rightarrow X\ D$

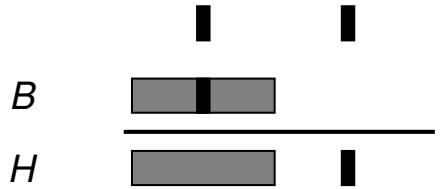
$A \rightarrow Y\ E$

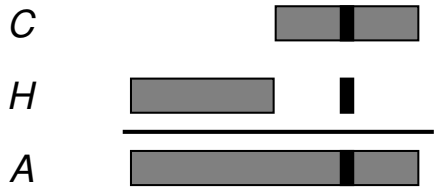
$O(n^{r+1}) \rightarrow O(n^3)$

(Chomsky Normal Form)

Rule factorization for bilexicalized parsing





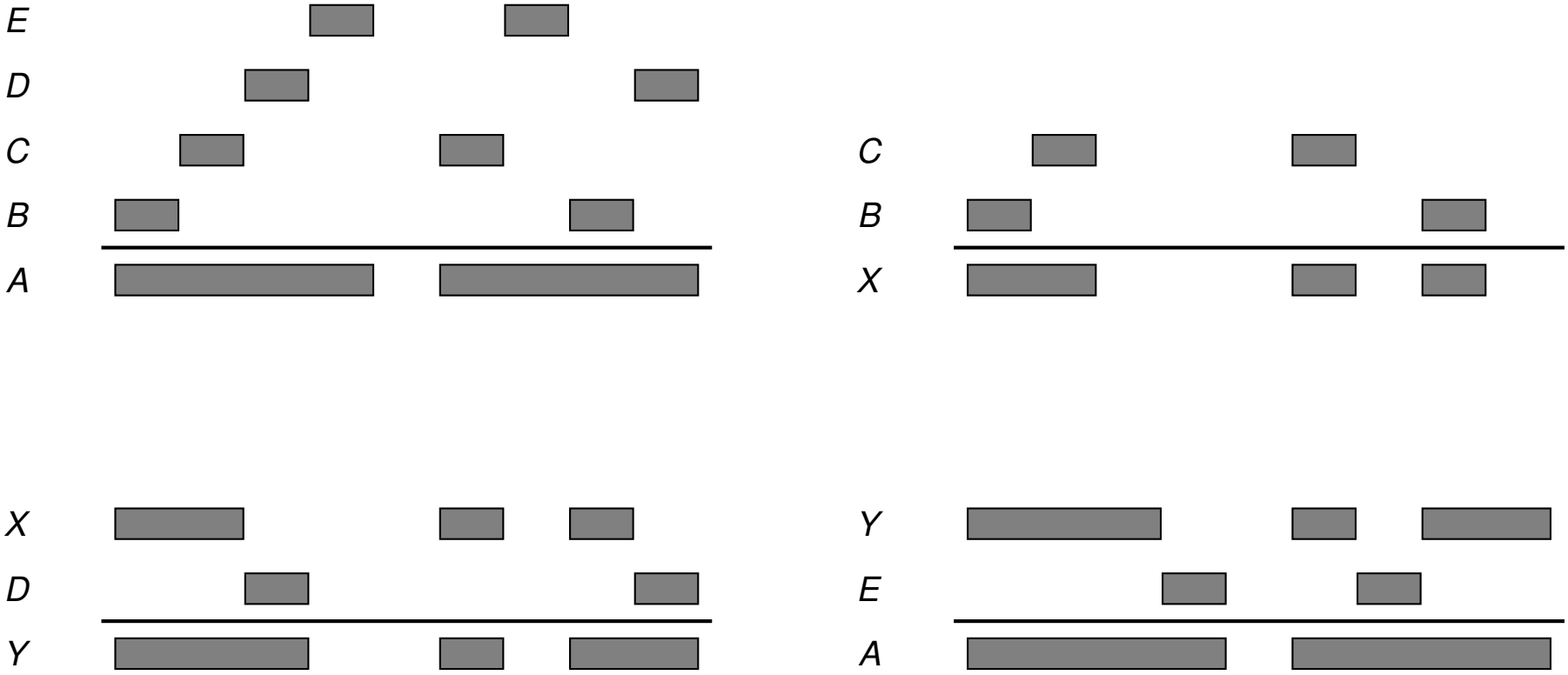


$O(n^5) \rightarrow O(n^4)$

(Eisner and Satta, 1999)

Synchronous Context-Free Grammar (SCFG)

$$A \rightarrow B^{(1)}\ C^{(2)}\ D^{(3)}\ E^{(4)},\ \ C^{(2)}\ E^{(4)}\ B^{(1)}\ D^{(3)}$$



Grammar Factorization

- Factor rule into intermediate rules
- Lowers parsing complexity
- Considers rules independently

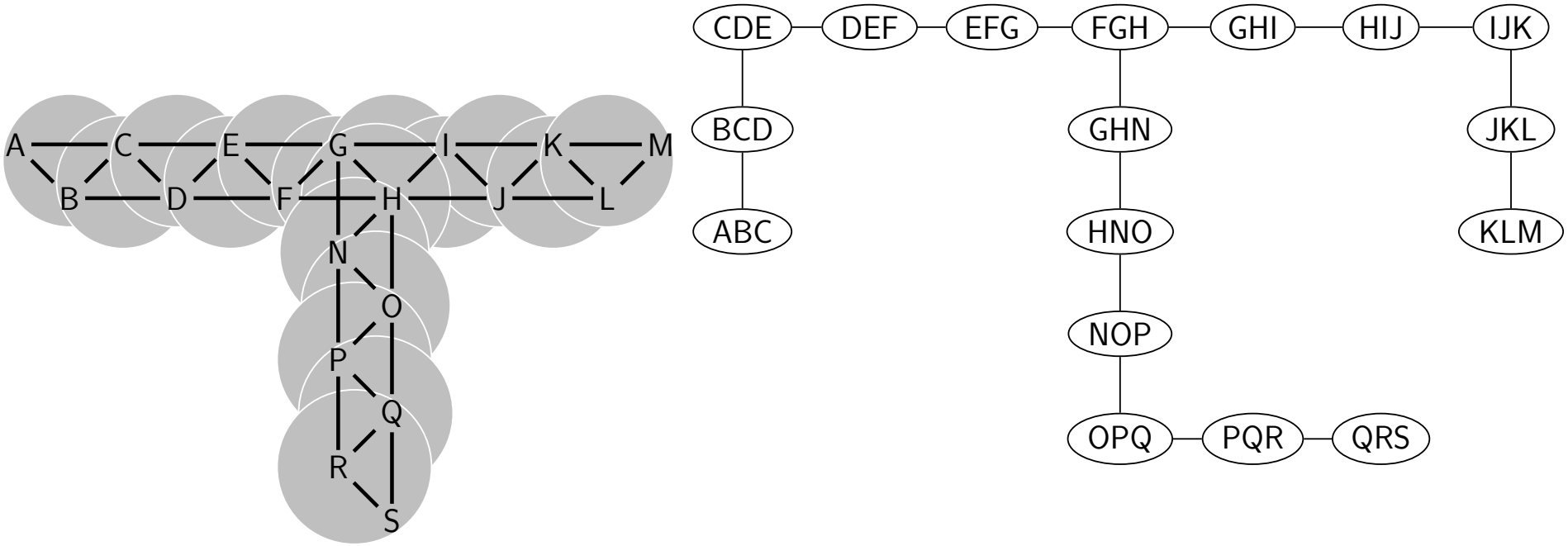
Grammar Factorization

- CFG: $O(n^{r+1}) \rightarrow O(n^3)$
- Bilex: $O(n^5) \rightarrow O(n^4)$
- SCFG: $O(n^{2r+2}) \rightarrow O(n^{cr})$

Goal: find best factorization for large rules:

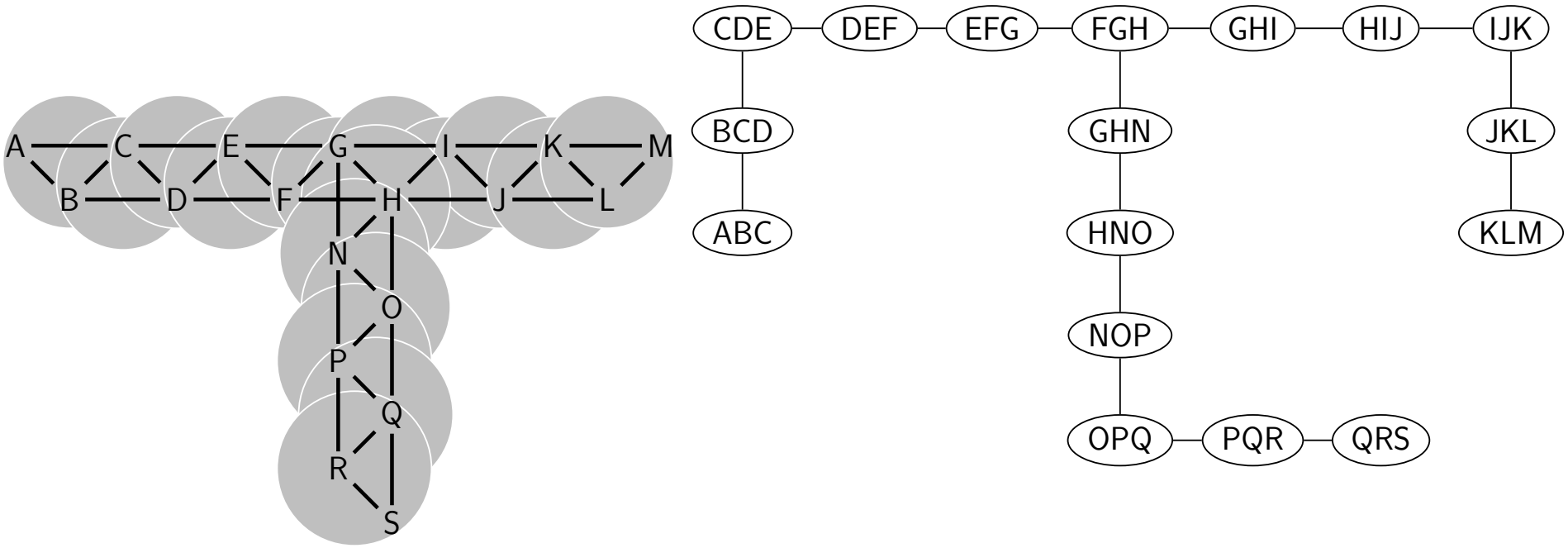
- Machine translation (Galley et al., 2004)
- Non-projective parsing (Kuhlman and Nivre, 2006; Gildea, 2010)

Tree Decomposition



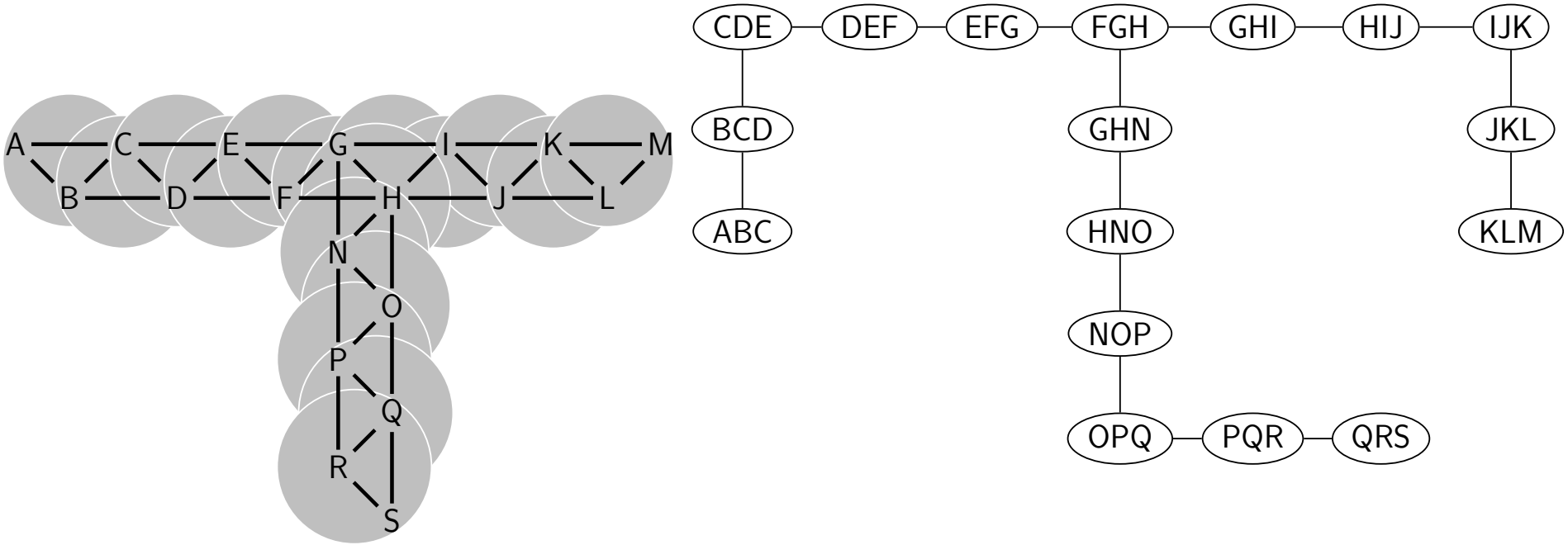
- every edge in some cluster
- clusters containing any vertex are connected

Tree Decomposition for 3SAT



- variable in 3SAT $\rightarrow$ node in graph
- appear in same clause in 3SAT $\rightarrow$ edge in graph

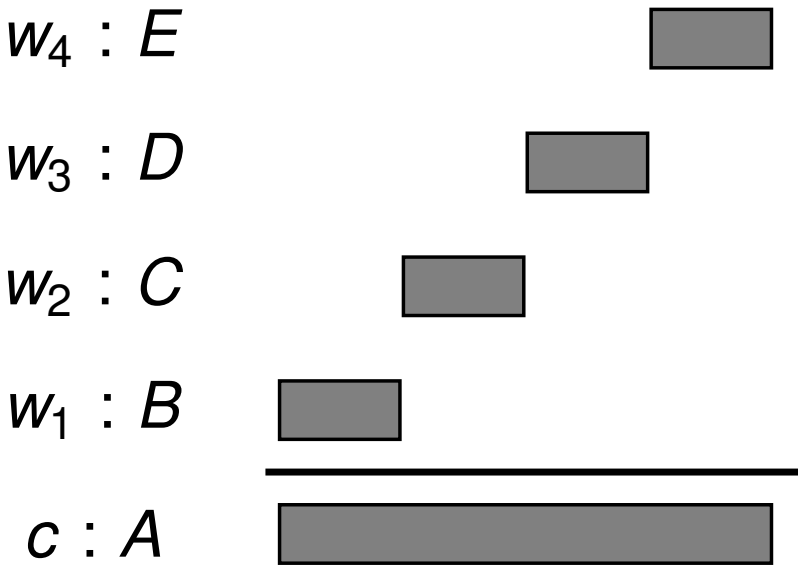
Tree Decomposition for Dynamic Programming

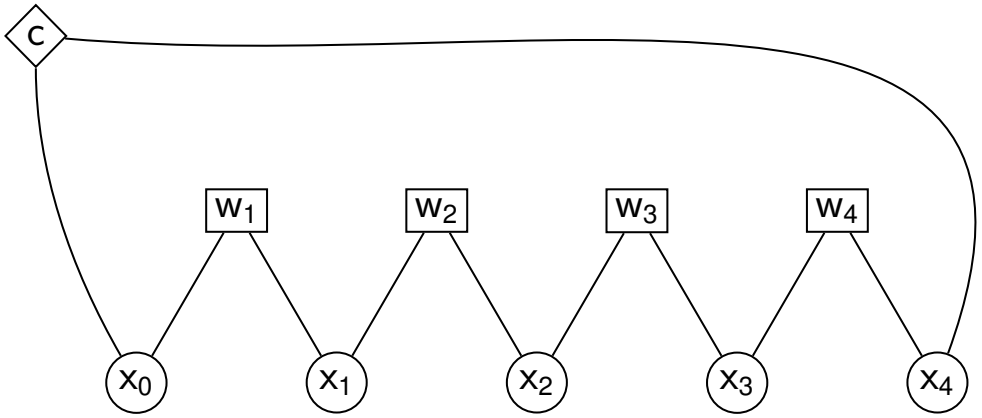


- every edge in some cluster $\rightarrow$ can apply constraint in cluster
- clusters containing any vertex are connected $\rightarrow$ variable has consistent value in solution

Factor Graph for CFG

$A \rightarrow B\ C\ D\ E$

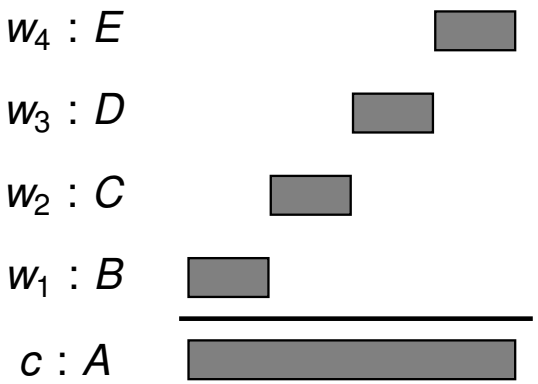




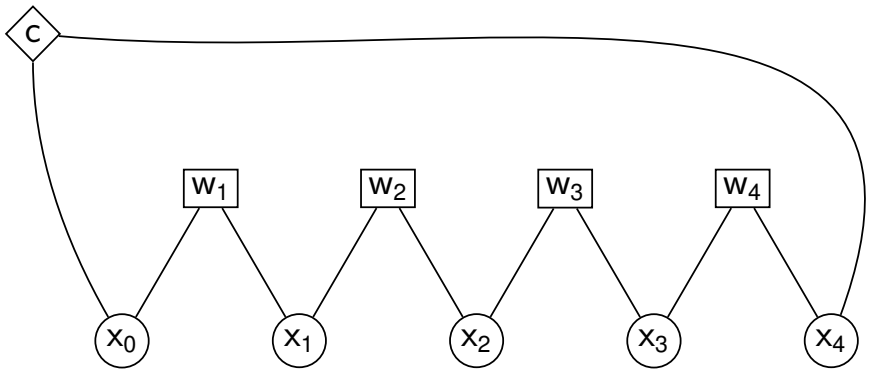
(Gildea, Computational Linguistics, 2010)

CFG Binarization

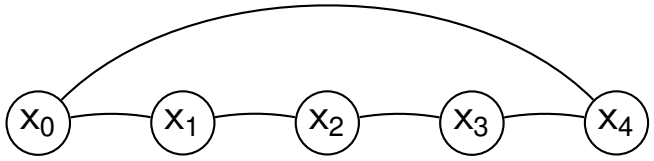
Deduction Rule:



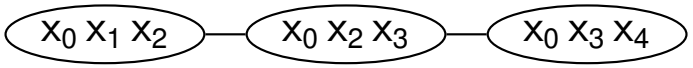
Factor Graph:



Dependency Graph:

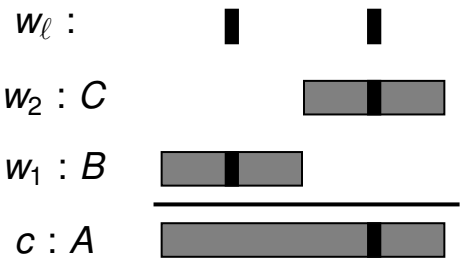


Tree Decomposition:

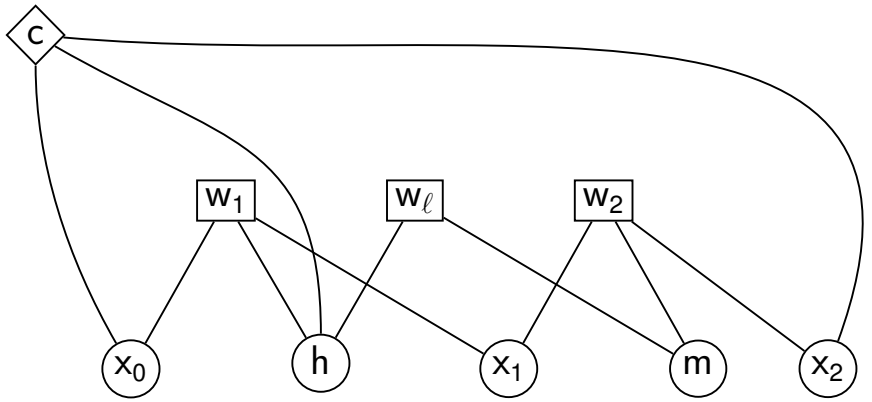


Bilexicalized Parsing

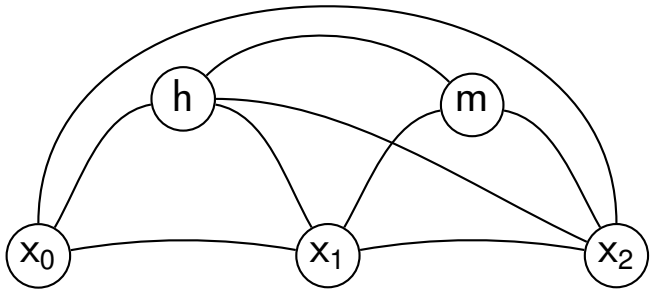
Deduction Rule:



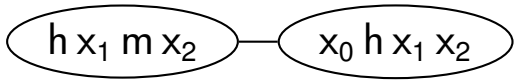
Factor Graph:



Dependency Graph:

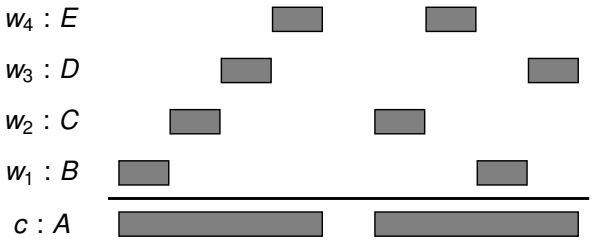


Tree Decomposition:

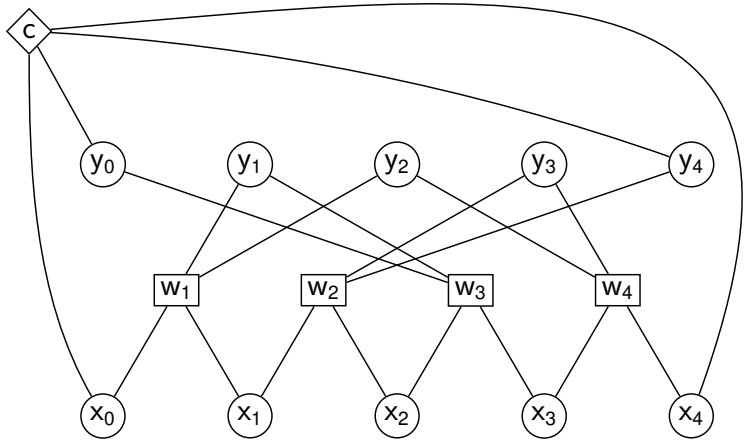


SCFG Parsing

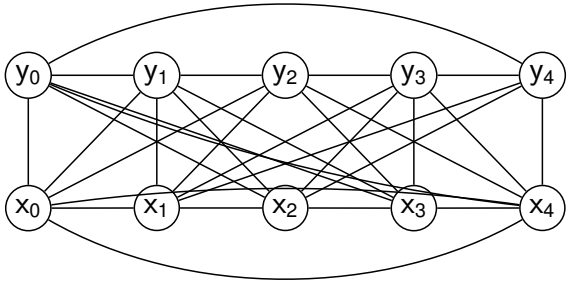
Deduction Rule:



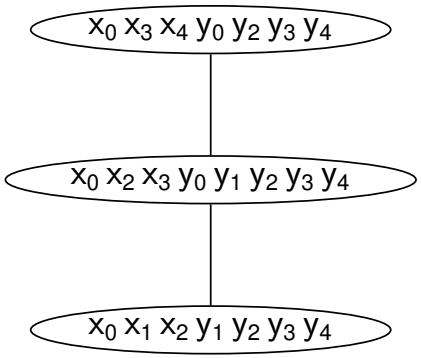
Factor Graph:



Dependency Graph:



Tree Decomposition:



Treewidth

- *Width* of tree decomposition is size of largest cluster
- Treewidth **tw**(G) is minimum width of any tree decomposition of G
- Optimal parsing complexity = treewidth

Complexity

- Treewidth for general graphs is NP-complete

  (Arnborg et al., 1987)

- Simple algorithm is $O(\ell^{k+2})$ for ℓ vertices and treewidth k
- Parsing with original grammar is $O(n^{\ell})$
- Treewidth is $O(\ell)$ for fixed k (Bodlaender, 1996)
- Efficient algorithms for LCFRS?

Linear Context-Free Rewriting Systems

LCFRS generalizes CFG, TAG, CCG, SCFG, STAG.

Productions $p \in P$ take the form:

$$p : A \rightarrow g(B_1, B_2, \ldots, B_r)$$

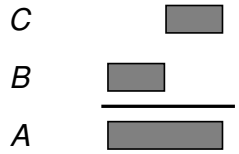
where $A, B_1, \ldots B_r \in V_N$, and g is a **linear, non-erasing** function

$$g(\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)} \rangle, \ldots, \langle x_{1,1}, \ldots, x_{1,\varphi(B_r)} \rangle) = \langle t_1, \ldots, t_{\varphi(A)} \rangle$$

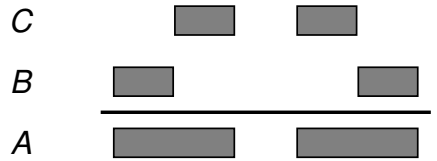
(Vijay-Shankar et al. ACL 1987)

Linear Context-Free Rewriting Systems

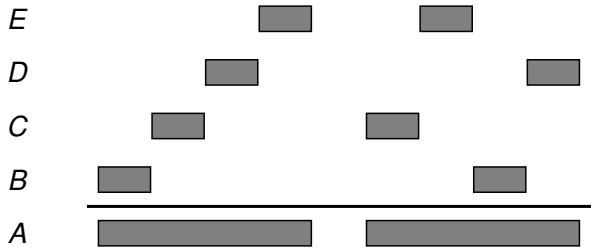
CFG: YES



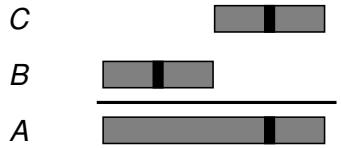
TAG: YES



SCFG: YES



Bilex: NO

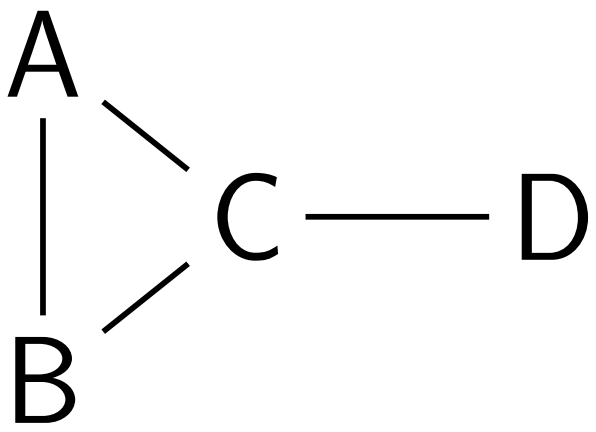


Treewidth for LCFRS

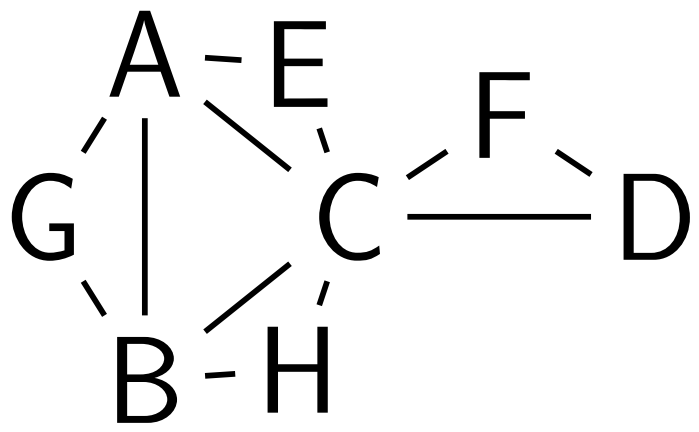
- Alg. for treewidth of LCFRS rule would imply $4\Delta(G)$-approximation algorithm for treewidth for general graphs.
- Proof strategy:
  - Take arbitrary graph G
  - From G, construct LCFRS production P with dependency graph G''
  - Show relation between $\mathbf{tw}(G)$ and $\mathbf{tw}(G'')$

Edge Doubling

G_{ex}:

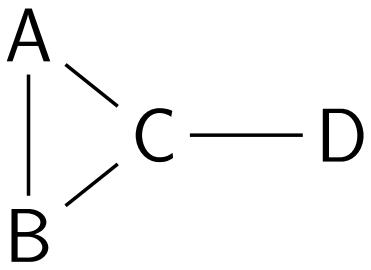


G'_{ex}:

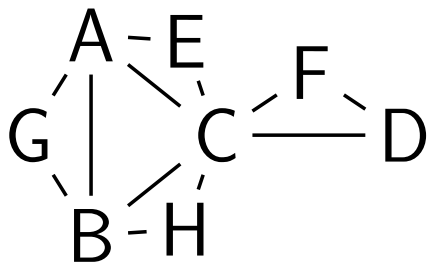


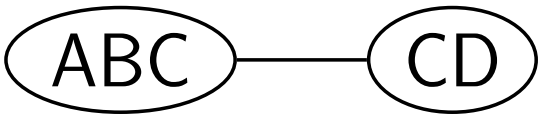
Tree decompositions of G_{ex} and G'_{ex}

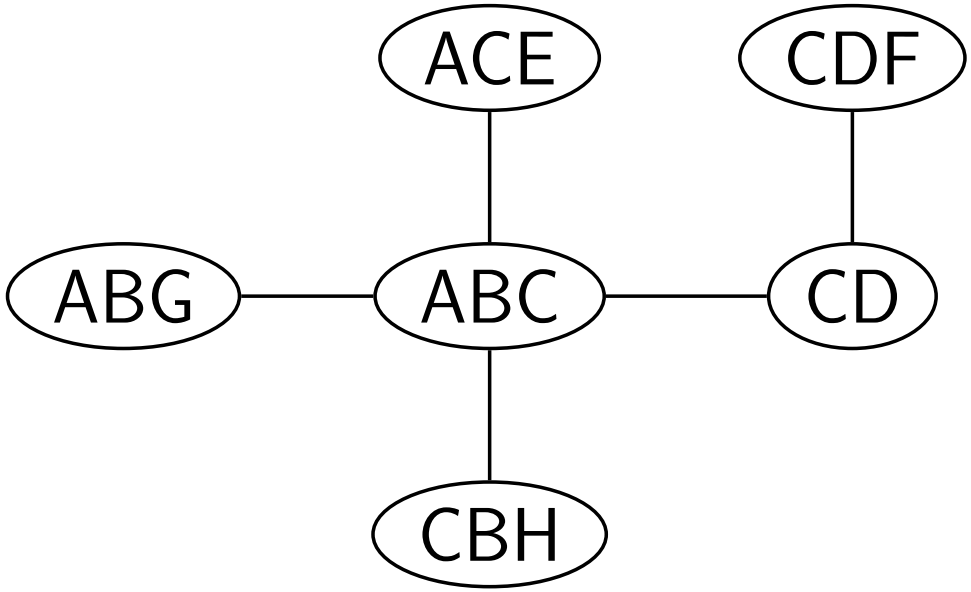
G_{ex}:

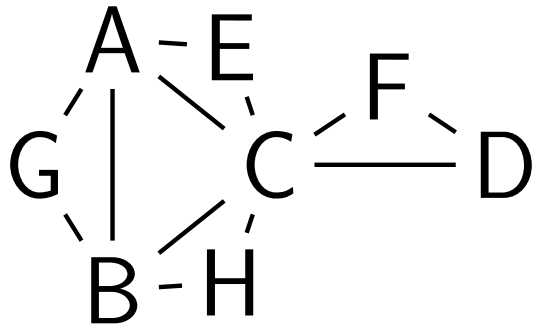


G'_{ex}:









Eulerian tour: $\langle A, B, C, D, F, C, E, A, G, B, H, C, A \rangle$

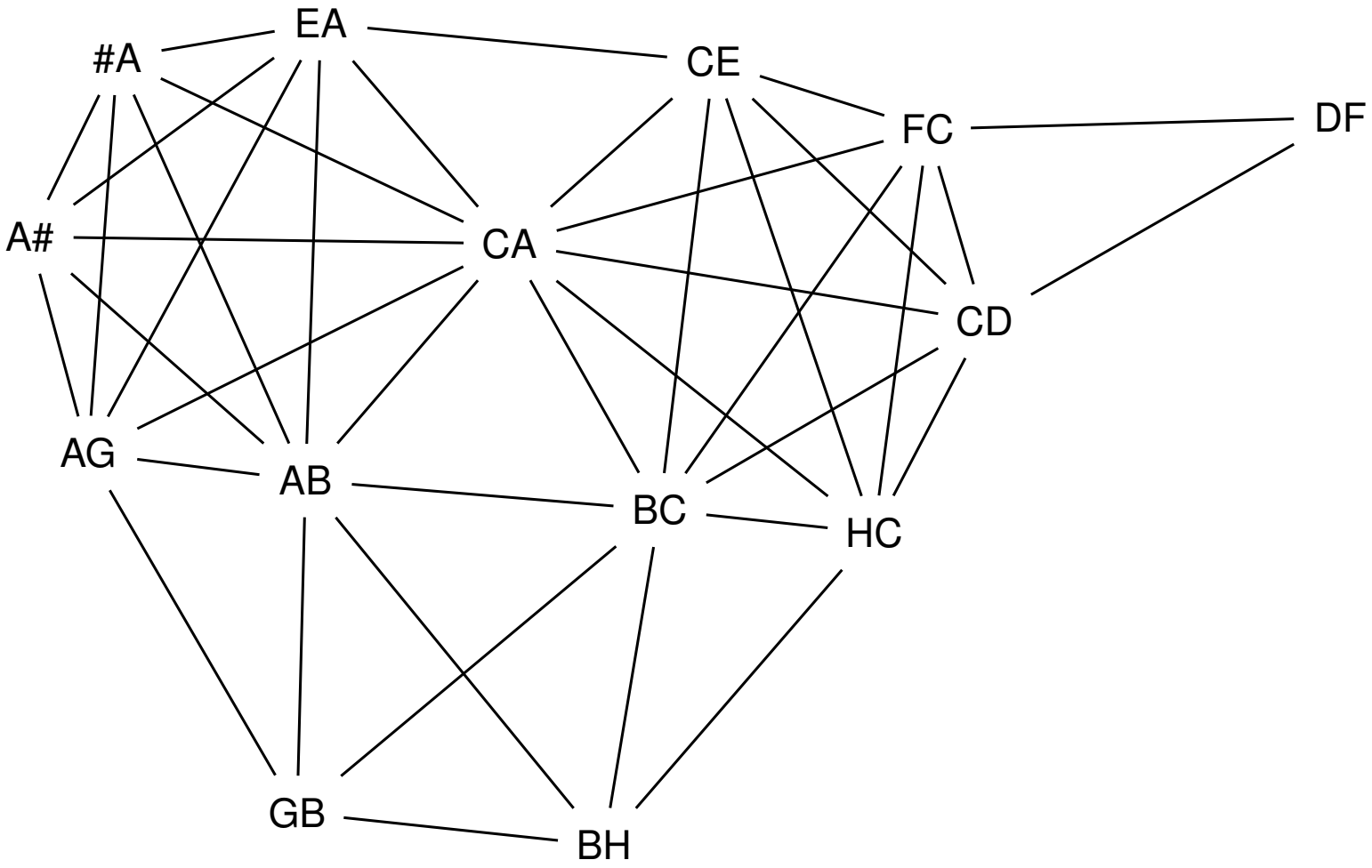
$$P_{ex} : X \rightarrow g_{ex}(A, B, C, D, E, F, G, H)$$

$$g_{ex}(\langle s_{A,1}, s_{A,2}, s_{A,3} \rangle, \langle s_{B,1}, s_{B,2} \rangle, \langle s_{C,1}, s_{C,2}, s_{C,3} \rangle,$$

$$\langle s_{D,1} \rangle, \langle s_{E,1} \rangle, \langle s_{F,1} \rangle, \langle s_{G,1} \rangle, \langle s_{H,1} \rangle) =$$

$$\langle s_{A,1} s_{B,1} s_{C,1} s_{D,1} s_{F,1} s_{C,2} s_{E,1} s_{A,2} s_{G,1} s_{B,2} s_{H,1} s_{C,3} s_{A,3} \rangle$$

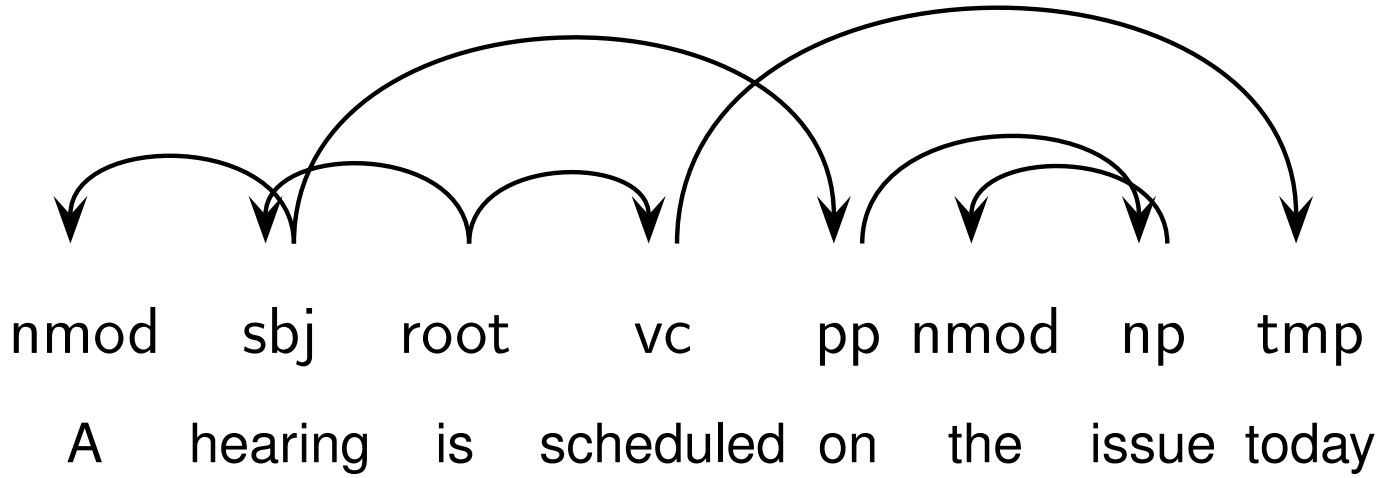
Dependency Graph



Approximation Alg.

- Map tree decomp. of G'' to tree decomp. G of treewidth SOL.
- $\mathbf{tw}(G) \leq SOL \leq 4\Delta(G)\mathbf{tw}(G)$
- Implies $4\Delta(G)$ approximation alg. for treewidth
- Best known approximation algs:
  - $O(\log k)$ (Amir, 2001)
  - $O(\sqrt{\log k})$ (Feige at al., 2005)

Dependency Treebank Experiments



$$\text{nmod} \rightarrow g_1 \qquad g_1 = \langle\, A \,\rangle$$

$$\text{sbj} \rightarrow g_2(\text{nmod}, \text{pp}) \qquad g_2(\langle x_{1,1} \rangle, \langle x_{2,1} \rangle) = \langle x_{1,1}\ \mathit{hearing}\ , x_{2,1} \rangle$$

$$\text{root} \rightarrow g_3(\text{sbj}, \text{vc}) \qquad g_3(\langle x_{1,1}, x_{1,2} \rangle, \langle x_{2,1}, x_{2,2} \rangle) = \langle x_{1,1}\ \mathit{is}\ x_{2,1} x_{1,2} x_{2,2} \rangle$$

$$\text{vc} \rightarrow g_4(\text{tmp}) \qquad g_4(\langle x_{1,1} \rangle) = \langle\, \mathit{scheduled}\ , x_{1,1} \rangle$$

$$\text{pp} \rightarrow g_5(\text{np}) \qquad g_5(\langle x_{1,1} \rangle) = \langle\, \mathit{on}\ x_{1,1} \rangle$$

$$\text{nmod} \rightarrow g_6 \qquad g_6 = \langle\, \mathit{the} \,\rangle$$

$$\text{np} \rightarrow g_7(\text{nmod}) \qquad g_7(\langle x_{1,1} \rangle) = \langle x_{1,1}\ \mathit{issue} \,\rangle$$

$$\text{tmp} \rightarrow g_8 \qquad g_8 = \langle\, \mathit{today} \,\rangle$$

Dependency Treebank Experiments

Kuhlmann and Nivre (ACL 2006) define “mildly non-projective dependency structures”.

Gomez-Rodriguez et al. (ACL 2009) define “mildly ill-nested dependency structures” parsed in $O(n^{3k+4})$.

Treebank Parsing Complexity

complexity	arabic	czech	danish	dutch	german	port	swedish
20							1
18							1
16							1
15							1
13							1
12						2	3
11					1	1	1
10		2			6	16	3
9					7	4	1
8		4		7	129	65	10
7		3		12	89	30	18
6		178	11	362	1811	492	59
5	48	1132	93	411	1848	172	201
4	250	18269	1026	6678	18124	2643	1736
3	10942	265202	18306	39362	154948	41075	41245

(Gildea, NAACL, 2010)

Conclusion

- Grammar Factorization = Tree Decomposition
- Parsing Complexity = Treewidth
- Optimal factorization is NP-complete
- Optimal factorization is possible when parsing is possible
- Optimal factorization of LCFRS implies $4\Delta(G)$ approx alg. for treewidth