Dynamic programming for parsing and estimation of Stochastic LFGs

Mark Johnson

Brown University

LFG'02, Athens

Acknowledgements: Stuart Geman, Stefan Riezler

Talk outline

- Why stochastic grammars?
- Stochastic LFGs
  - Selecting the optimal (most likely) parse
  - (Relationship to Optimality Theory)
  - Exponential distributions
  - Learning stochastic LFGs
- Dynamic programming for stochastic LFGs
  - Maxwell-Kaplan contexted representations
  - Property locality
  - Avoiding enumerating all parses

Two problems of non-statistical CL

1. *Ambiguity explodes combinatorially*

   (162) *Even though it's possible to scan using the Auto Image Enhance mode, it's best to use the normal scan mode to scan your documents.*

   - Refining the grammar is often self-defeating
     $\Rightarrow$ splits states $\Rightarrow$ makes the problem worse!
   - Preference information guides parser to correct analysis

2. *Requiring grammaticality leads to non-robustness*

   - Ungrammatical sentences are often comprehensible

Optimization and Statistics

- An *optimization-based approach* can solve both problems

  Robustness: Every interpretable string receives *at least one parse*

  Ambiguity: Parses are ranked, parser returns *highest ranked parse*

- Why statistics?
  - Parsing and acquisition are *inference problems*
  - Statistics is the study of *inference under uncertainty*

Linguistics and statistical parsing

- Statistical parsers are *not* “linguistics-free”
  - The training data consists of syntactic parses
  - Possible features specified manually
- *What is the most effective way to import useful linguistic knowledge?*
  - *manually specify* linguistic representations (LFG)
  - *manually specify* features
  - *learn* feature weights from training data

How to select the optimal parse?

- There are *many features* that might be relevant
  - Subcategorization preferences
  - Prefer argument over adjunct attachment
  - Selectional preferences
  - Parallelism of coordinate structures
- How do we choose a parse when the features are contraditory?
  - Rank the features; use the highest ranked features (OT)
  - Weight each feature; use a weighted combination of features

Linear models of parse selection

- An LFG parser produces *parses* $\{y_1, \ldots, y_\ell\}$ of the sentence
- Each feature f_j is a *real-valued function*: $f_j(y)$ scores y
- Each feature f_j has a *real-valued weight* λ_j
- The *score* $S(y)$ of parse y is

$$S(y) \;=\; \lambda_1 f_1(y) + \ldots + \lambda_m f_m(y) \;=\; \sum_{j=1}^{m} \lambda_j f_j(y)$$

- The *highest scoring parse* is selected

$$\hat{y} \;=\; \operatorname*{argmax}_{y \in \{y_1, \ldots, y_\ell\}} S(y)$$

Why a *linear* combination of features?

- Intuition: No single feature alone reliably indicates best parse
- Linear combination of features is about the simplest possible
  - Each feature can be arbitrarily complex
  - Really a restriction on complexity of *learning*
- Used in virtually all other statistical models
- Has impeccable information-theoretic justification

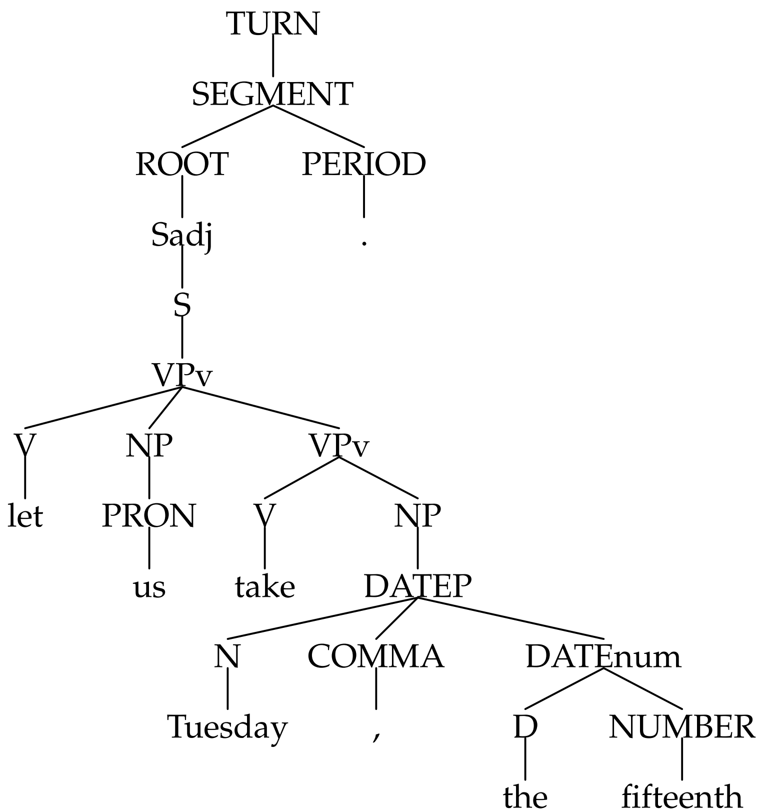
Aside: Simulating OT with a linear model

- For each OT constraint c introduce a feature f_c:

$$f_c(y) \; = \; \text{number of times } y \text{ violates } c$$

- Feature weight $\approx$ constraint ranking
  - Feature weights for OT constraints are always negative
  - OT: strict domination
  - Linear model: no strict domination
- If *number of constraint violations is bounded* then for every OT constraint ranking there are feature weights so that *the OT optimal parse is the highest linear scoring parse*

Sample LFG parse




```
SENTENCE_ID   BAC002_E

OBJ     ANIM   +
        CASE   ACC
        NUM    PL
        PERS   1
        PRED   PRO
        PRON-FORM   WE
        PRON-TYPE   PERS
      9
PASSIVE   –
PRED      LET⟨2,10⟩9
STMT-TYPE   IMPERATIVE
SUBJ    PERS   2
        PRED   PRO
        PRON-TYPE   NULL
      2
TNS-ASP   MOOD   IMPERATIVE

XCOMP   OBJ   ANIM   –
              APP    NTYPE   NUMBER   ORD
                             TIME   DATE
                     NUM    SG
                     PRED   fifteen
                     SPEC   SPEC-FORM   THE
                            SPEC-TYPE   DEF
              CASE   ACC
              GEND   NEUT
              NTYPE  GRAIN   COUNT
                     PROPER   DATE
                     TIME   DAY
              NUM    SG
              PERS   3
              PRED   TUESDAY
            13
        PASSIVE   –
        PRED   TAKE⟨9,13⟩
        SUBJ   [9]
10
```

Features

Rule features: For every non-terminal X, $f_X(y)$ is the number of times X occurs in c-structure of y

Attribute value features: For every attribute a and every atomic value v, $f_{a=v}(y)$ is the number of times the pair $a = v$ appears in y

Argument and adjunct features: For every grammatical function g, $f_g(y)$ is the number of times that g appears in y

Other features: Dates, times, locations; right branching; attachment location; parallelism in coordination; …

- Choice of features is a linguistic issue

Learning the feature weights

- Training data: sentences and their "correct" parses
- Parse each sentence to obtain its competitors
- Compute feature values for correct and competitor parses

	Correct parse's features	**All other parses' features**
sentence 1	$[1, 3, 2]$	$[2, 2, 3]$ $[3, 1, 5]$ $[2, 6, 3]$
sentence 2	$[7, 2, 1]$	$[2, 5, 5]$
sentence 3	$[2, 4, 2]$	$[1, 1, 7]$ $[7, 2, 1]$

- Choose feature weights so that correct parses are most often optimal

Estimating the feature weights

- Classical discriminative learning $\Rightarrow$ many algorithms
- We used *maximum conditional likelihood estimation* of a *log linear model* (a.k.a. exponential, MaxEnt, Harmony, ...)
  - Conditional probability of a parse y of a sentence x

$$\Pr_\lambda(y|x) \;=\; 1/Z(x)\exp(S(y))$$

  - Conditional likelihood of data $D = (x_1, y_1), \ldots, (x_n, y_n)$

$$L_D(\lambda) \;=\; \prod_{i=1}^{n} \Pr_\lambda(y_i|x_i)$$

  - Maximum conditional likelihood estimate of λ

$$\hat{\lambda} \;=\; \operatorname*{argmax}_{\lambda} L_D(\lambda)$$

Estimating feature weights *efficiently*

- No known analytical solution for $\hat{\lambda}$
- Most other estimation techniques also don't have analytical solutions

$\Rightarrow$ Use standard iterative numerical optimization techniques

$\Rightarrow$ Repeated reparsing of training data

- Modern parsers (e.g., XLE) produce *packed representations* of parses
- Can we work directly from these packed representations, i.e., avoid unpacking? *YES**
  – Related work: Miyao and Tsuji (2002) "Maximum Entropy Estimation for Feature Forests" HLT

Product form of log linear model

- Reparameterize feature weights: $\theta_j = \exp(\lambda_j)$

$$W(y) \;=\; \exp(S(y)) \;=\; \exp(\sum_{j=1}^{m} \lambda_j f_j(y)) \;=\; \prod_{j=1}^{m} {\theta_j}^{f_j(y)}$$

- Optimal parse $\hat{y}$

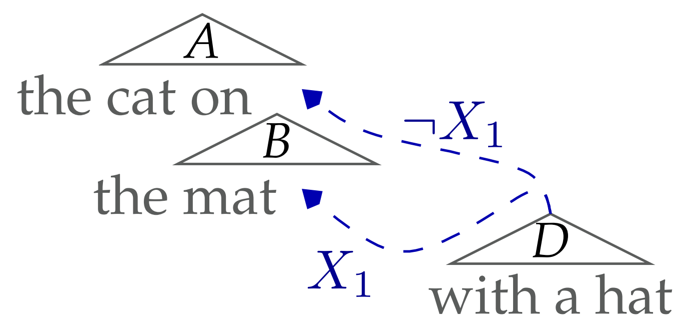
$$\hat{y} \;=\; \operatorname*{argmax}_{y \in \{y_1,\ldots,y_\ell\}} S(y) \;=\; \operatorname*{argmax}_{y \in \{y_1,\ldots,y_\ell\}} W(y)$$

- Aside: Every PCFG is a log linear model in product form
  - The production probabilities are the θ_j
  - The probability of a parse is $W(y)$

Maxwell and Kaplan packed parses

- A parse y consists of set of fragments $\xi \in y$
- A fragment is in a parse when its *context function* is true
- Context functions are functions of zero or more *context variables*
- The variable assignment must satisfy "not no-good" functions
- Each parse is identified by a *unique context variable assignment*

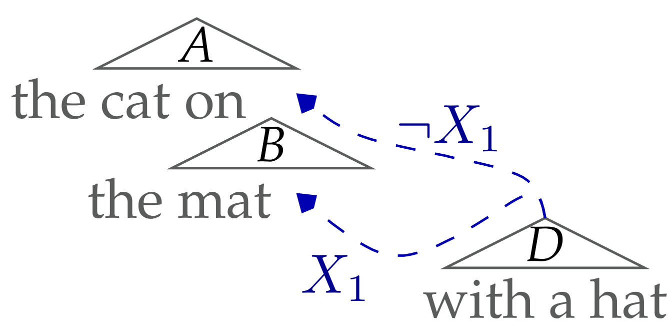
$$\begin{aligned} \xi &= \text{“\textit{the cat on the mat}”} \\ \xi_1 &= \text{“\textit{with a hat}”} \\ X_1 &\rightarrow \text{“attach } D \text{ to } B\text{”} \\ \neg X_1 &\rightarrow \text{“attach } D \text{ to } A\text{”} \end{aligned}$$



Feature locality

- Features must be *local* to fragments: $f_j(y) = \sum_{\xi \in y} f_j(\xi)$
- May require changes to LFG to make all features local

$$\begin{aligned} \xi &= \text{“\textit{the cat on the mat}”} \\ \xi_1 &= \text{“\textit{with a hat}”} \\ X_1 &\rightarrow \text{“attach } D \text{ to } B\text{”} \ \wedge\ (\xi_1 \text{ ATTACH}) = \text{LOW} \\ \neg X_1 &\rightarrow \text{“attach } D \text{ to } A\text{”} \ \wedge\ (\xi_1 \text{ ATTACH}) = \text{HIGH} \end{aligned}$$



Feature locality decomposes $W(y)$

- Feature locality: the weight of a parse is the product of weights of its fragments

$$\begin{aligned} W(y) &= \prod_{\xi \in y} W(\xi), \qquad \text{where} \\ W(\xi) &= \prod_{j=1}^{m} \theta_j^{f_j(\xi)} \end{aligned}$$

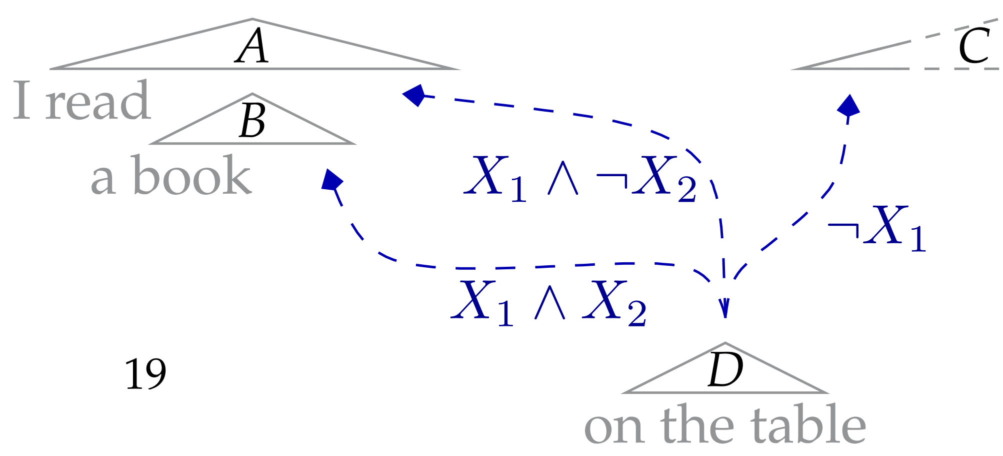
$$\begin{aligned} W(\xi &= \textit{“the cat on the mat”}) \\ W(\xi_1 &= \textit{“with a hat”}) \end{aligned}$$

$$\begin{aligned} X_1 &\to W\,(\text{“attach } D \text{ to } B\text{”} \ \wedge \ (\xi_1 \ \text{ATTACH}) = \text{LOW} \ ) \\ \neg X_1 &\to W\,(\text{“attach } D \text{ to } A\text{”} \ \wedge \ (\xi_1 \ \text{ATTACH}) = \text{HIGH} \ ) \end{aligned}$$

Not No-goods

- "Not no-goods" identify the variable assignments that correspond to parses

$$
\begin{aligned}
\xi &= \textit{“I read a book”} \\
\xi_1 &= \textit{“on the table”} \\
X_1 \wedge X_2 &\rightarrow \text{“attach } D \text{ to } B\text{”} \\
X_1 \wedge \neg X_2 &\rightarrow \text{“attach } D \text{ to } A\text{”} \\
\neg X_1 &\rightarrow \text{“attach } D \text{ to } C\text{”} \\
& X_1 \vee X_2
\end{aligned}
$$



Identify parses with variable assignments

- *Each variable assignment uniquely identifies a parse*
- For a given sentence w, let $W'(x) = W(y)$ where y is the parse identified by x

$\Rightarrow$ Argmax/sum/expectations over parses can be computed over context variables instead

Most likely parse: $\hat{x} = \operatorname{argmax}_x W'(x)$

Partition function: $Z(w) = \sum_x W'(x)$

Expectation:* $E[f_j|w] = \sum_x f_j(x) W'(x)/Z(w)$

W' is a product of functions of X

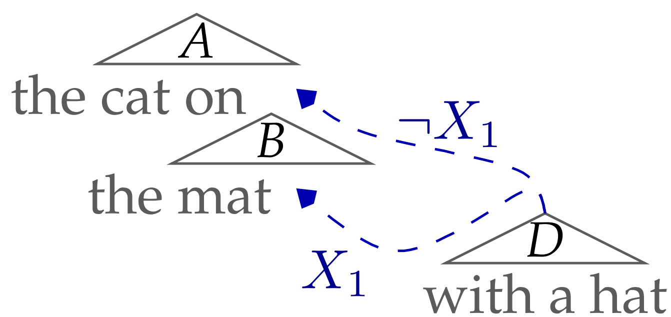
- Write $W(X) = \prod_{A \in \mathcal{A}} A(X)$, where:
  - Each line $\alpha(X) \to \xi$ introduces a term $W(\xi)^{\alpha(X)}$
  - A "not no-good" $\eta(X)$ introduces a term $\eta(X)$

$$
\begin{array}{ccccc}
 & \vdots & & & \vdots \\
\alpha(X) & \to & \xi & \times & W(\xi)^{\alpha(X)} \\
 & \vdots & & \times & \vdots \\
 & & \eta(X) & \times & \eta(X) \\
 & \vdots & & \times & \vdots
\end{array}
$$

$\Rightarrow$ W' is a *Markov Random Field* over the context variables X

W' is a product of functions of X

$$
\begin{aligned}
W'(X_1) \;=\; & \quad W(\xi = \textit{“the cat on the mat”}) \\
& \times\; W(\xi_1 = \textit{“with a hat”}) \\
& \times\; W\left(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{low}\right)^{X_1} \\
& \times\; W\left(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{high}\right)^{\neg X_1}
\end{aligned}
$$



Product expressions and graphical models

- MRFs are products of terms, each of which is a function of (a few) variables
- Graphical models provide *dynamic programming algorithms* for Markov Random Fields (MRF) (Pearl 1988)
- These algorithms implicitly *factorize the product*
- They generalize the Viterbi and Forward-Backward algorithms to arbitrary graphs (Smyth 1997)

$\Rightarrow$ Graphical models provide dynamic programming techniques for parsing and training Stochastic LFGs

Factorization example

$$
\begin{aligned}
W'(X_1) \;=\; & \quad W(\xi \;=\; \text{“\textit{the cat on the mat}”}) \\
\times & \quad W(\xi_1 \;=\; \text{“\textit{with a hat}”}) \\
\times & \quad W\left(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1 \text{ ATTACH}) = \text{LOW}\right)^{X_1=1} \\
\times & \quad W\left(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1 \text{ ATTACH}) = \text{HIGH}\right)^{X_1=0}
\end{aligned}
$$

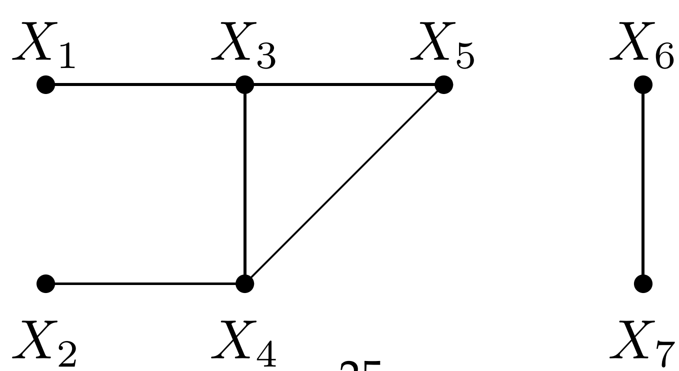
$$
\begin{aligned}
\max_{X_1} W'(X_1) \;=\; & \quad W(\xi \;=\; \text{“\textit{the cat on the mat}”}) \\
\times & \quad W(\xi_1 \;=\; \text{“\textit{with a hat}”}) \\
\times & \quad \max_{X_1} \begin{pmatrix} W\left(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1 \text{ ATTACH}) = \text{LOW}\right)^{X_1=1}, \\ W\left(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1 \text{ ATTACH}) = \text{HIGH}\right)^{X_1=0} \end{pmatrix}
\end{aligned}
$$

Dependency structure graph $G_{\mathcal{A}}$

$$\hat{x} \;=\; \operatorname*{argmax}_x W(x) \;=\; \operatorname*{argmax}_x \prod_{A \in \mathcal{A}} A(x)$$

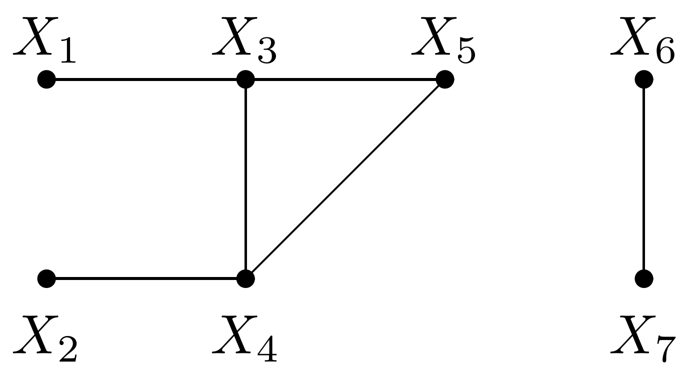
- $G_{\mathcal{A}}$ is the *dependency graph* for $\mathcal{A}$
  - context variables X are vertices of $G_{\mathcal{A}}$
  - $G_{\mathcal{A}}$ has an edge (X_i, X_j) if both are arguments of some $A \in \mathcal{A}$

$$A(X) = a(X_1, X_3) b(X_2, X_4) c(X_3, X_4, X_5) d(X_4, X_5) e(X_6, X_7)$$



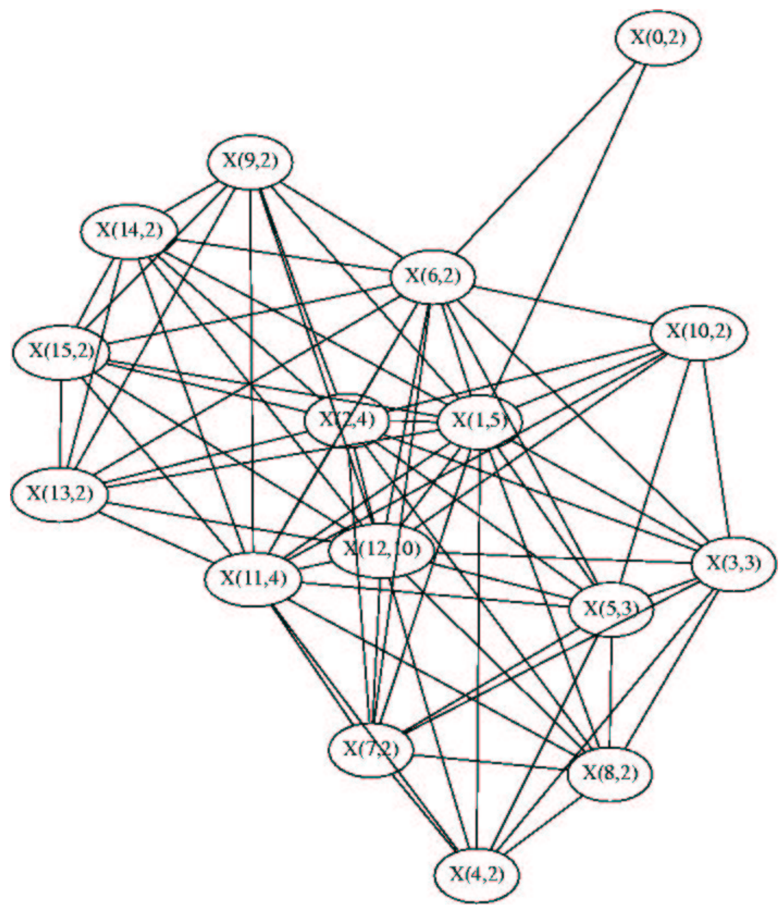
Graphical model computations

$$
\begin{aligned}
\hat{x} &= \operatorname{argmax}_x a(x_1,x_3)b(x_2,x_4)c(x_3,x_4,x_5)d(x_4,x_5)e(x_6,x_7) \\
W(\hat{x}) &= \max_x a(x_1,x_3)b(x_2,x_4)c(x_3,x_4,x_5)d(x_4,x_5)e(x_6,x_7) \\
W_1(x_3) &= \max_{x_1} a(x_1,x_3) \\
W_2(x_4) &= \max_{x_2} b(x_2,x_4) \\
W_3(x_4,x_5) &= \max_{x_3} c(x_3,x_4,x_5)W_1(x_3) \\
W_4(x_5) &= \max_{x_4} d(x_4,x_5)W_2(x_4)W_3(x_4,x_5) \\
W_5 &= \max_{x_5} W_4(x_5) \\
W_6(x_7) &= \max_{x_6} e(x_6,x_7) \\
W_7 &= \max_{x_7} F_6(x_7) \\
W(\hat{x}) &= W_5 W_7
\end{aligned}
$$



Graphical model for Homecentre example

Use a damp, lint-free cloth to wipe the dust and dirt buildup from the scanner plastic window and rollers.



Computational complexity

- Polynomial in m = the *maximum number of variables in the dynamic programming functions* $\geq$ the number of variables in any function A
- m depends on the ordering of variables (and G)
- Finding the variable ordering that minimizes m is NP-complete, but there are good heuristics

$\Rightarrow$ Worst case exponential (no better than enumerating the parses), but average case might be much better

  – Much like LFG parsing complexity

Conclusion

- It is possible to directly compute the statistics for parsing and estimation from Maxwell and Kaplan packed parses
- Features must be local to parse fragments
  - May require adding features to the grammar
- Computational complexity is worst-case exponential, average case?
- Makes available techniques for graphical models to Stochastic Lexical-Functional Grammar
  - MCMC and other sampling techniques

Future directions

- Reformulate "hard" LFG grammatical constraints as "soft" stochastic features
  - Underlying LFG permits all possible structural combinations
  - Grammatical constraints reformulated as stochastic features
- Is this computation tractable?
- Relationship with Dynamic Programming in Optimality Theory
  - Can these techniques be extended to OT LFG?