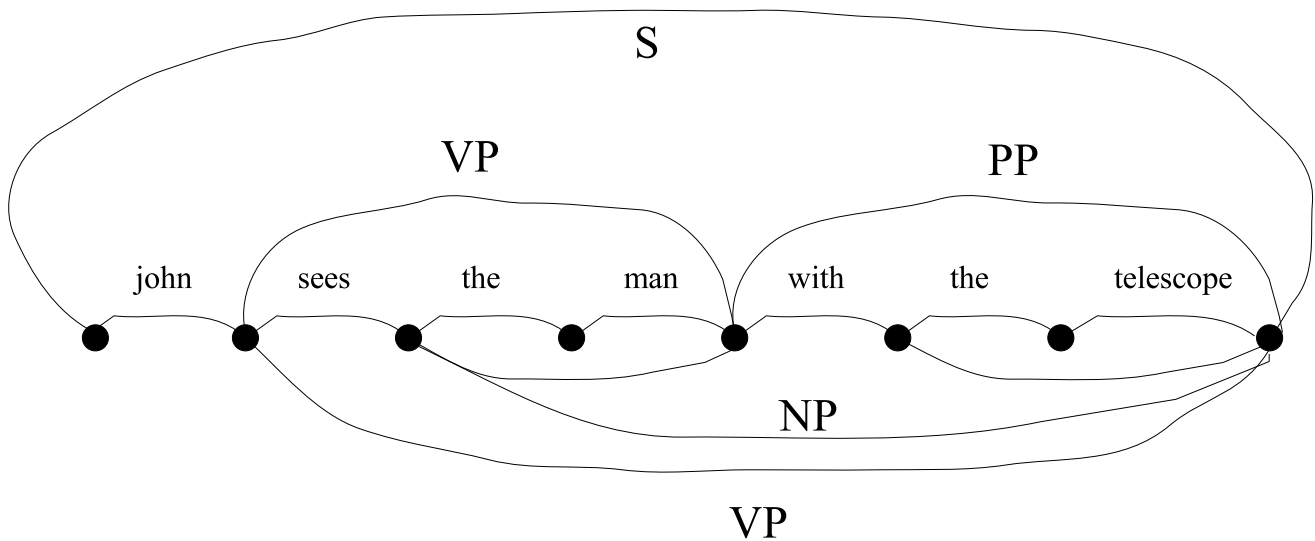


Control Strategies
for Parsing with
Freer Word-Order Languages

Gerald Penn
University of Toronto
gpenn@cs.toronto.edu
<http://www.cs.toronto.edu/~gpenn>

25 August, 2008
Charles University

Chart Parsing



- Simple tabulation method for CFG string recognition
- Can build bottom-up or top-down
- Table entries: **edge(Cat,Left,Right)**

Freer Word-Order Languages

Conjecture: Freer Word-Order (FWO) languages are the languages for which there is no known phrase structure grammar in which the rules:

- are context-independent,
- are semantically “transparent,” and
- tightly couple grammatical function assignment with linear constituent order.

We’d like to use standard chart parsers with FWO grammars, but then either:

- we must exponentially expand their rule systems into a classical context-independent rewriting system [Barton et al., 1987], or
- we must generalize the assumption that edges are characterizable by a left node, right node and category triple (*contiguous subspans*).

Freer Word-Order Languages

Conjecture: Freer Word-Order (FWO) languages are the languages for which there is no known phrase structure grammar in which the rules:

- **are context-independent,**
- **are semantically “transparent,”** and
- **tightly couple grammatical . . .**

We'd like to use standard chart parsers with FWO grammars, but then either:

- **we must exponentially expand . . .**
- **we must generalize the assumption that edges are characterizable by a left node, right node and category triple (*contiguous subspans*).**

Generalizing Parsing State

Use input-length bit vectors [Johnson, 1985; Reape, 1991; et al.] plus categories in edges ...

... but this isn't even half of the story:

1. danger of naïvely searching through 2^n states in a purely bottom-up approach:

- no goal-directedness,
- linear precedence constraints ($<$) can help, but almost without exception their scopes are:
 - not purely existential, e.g.: $\forall X.\forall Y.Y < X$
 - not clearly bounded, e.g.: all Y in the current NP, all Y in the current clause, etc.
 - not linearly projective, e.g.: NP $<$ VP, but what does the beginning of a VP look like?
 - specific to *trees*, not strings.

Generalizing Parsing State

Use input-length bit vectors [Johnson, 1985; Reape, 1991; et al.] plus categories in edges ...

... but this isn't even half of the story:

2. acute need for some top-down control, but:

- edge prediction: $\exists M.\mathbf{active}(L, \mathbf{Cat}, M)$ no longer works.
- edge subsumption: not as simple as comparing categories across identical bit vectors anymore.

3. the good news: classical ID schemata implicitly provide upper and lower bounds on the number of leaves in their subtrees (*yield*):

- even in the total absence of LP constraints,
- even in the abundant presence of recursion,
- conjecture: and they can make top-down parsing faster, even with classical CFGs.

Generalizing Parsing State

Use input-length bit vectors [Johnson, 1985; Reape, 1991; et al.] plus categories in edges ...

... but this isn't even half of the story:

2. acute need for some top-down control, but:

- **edge prediction**
- **edge subsumption.**

3. the good news: classical ID schemata implicitly provide upper and lower bounds on the number of leaves in their subtrees (*yield*):

- even in the total absence of LP constraints,
- even in the abundant presence of recursion,
- conjecture: and **they can be used to make top-down parsing faster, even with classical CFGs (work in progress).**

Edge Prediction

We need to distinguish between passive and active edges in our states [Penn & Haji-Abdolhosseini, 2003]:

- Passive: $\langle N, \text{UsedBV} \rangle$
- Active: $\langle N, \text{CanBV}, \text{ReqBV} \rangle$
 - (**C**)anBV: the set of words that *can* be used to derive this N.
 - (**R**)eqBV: the set of words that *must* be used to derive this N.
 - (**U**)sedBV: the set of words that *were* used to derive this N.

Edge Prediction

Using this distinction, the prediction steps are then (assuming no LP constraints, for simplicity):

- (Initial Prediction) $\langle N, C, R \rangle \implies \langle N_1, C, \phi \rangle$,
where:
 1. $N_0 \rightarrow N_1 \dots N_k$,
 2. $k > 1$, and
 3. $N = N_0$.
- (Subsequent Prediction)
- (Completion)

Edge Prediction

Using this distinction, the prediction steps are then (assuming no LP constraints, for simplicity):

- (Initial Prediction)
- (Subsequent Prediction) $\langle N, C, R \rangle \implies \langle N_{j+1}, C_j, \phi \rangle$
where:
 1. $N_0 \rightarrow N_1 \dots N_k$,
 2. $N = N_0$,
 3. $\langle N_1, C, \phi \rangle$ succeeded with U_1 ,
 - \vdots
 - $\langle N_j, C_{j-1}, \phi \rangle$ succeeded with U_j ,
 4. $k > 1$ and $1 \leq j < k - 1$, and
 5. $C_j = C \cap \bar{U}_1 \cap \dots \cap \bar{U}_j$.
- (Completion)

Edge Prediction

Using this distinction, the prediction steps are then (assuming no LP constraints, for simplicity):

- (Initial Prediction)
- (Subsequent Prediction)
- (Completion) $\langle N, C, R \rangle \implies \langle N_k, C_{k-1}, R_{k-1} \rangle$,
where:

1. $N_0 \rightarrow N_1 \dots N_k$,
2. $N = N_0$,
3. $\langle N_1, C, \phi \rangle$ succeeded with U_1 ,
- ⋮
- $\langle N_{k-1}, C_{k-2}, \phi \rangle$ succeeded with U_{k-1} ,
4. $C_{k-1} = C \cap \overline{U}_1 \cap \dots \cap \overline{U}_{k-1}$, and
5. $R_{k-1} = R \cap \overline{U}_1 \cap \dots \cap \overline{U}_{k-1}$.

Edge Prediction

Using this distinction, the prediction steps are then (assuming no LP constraints, for simplicity):

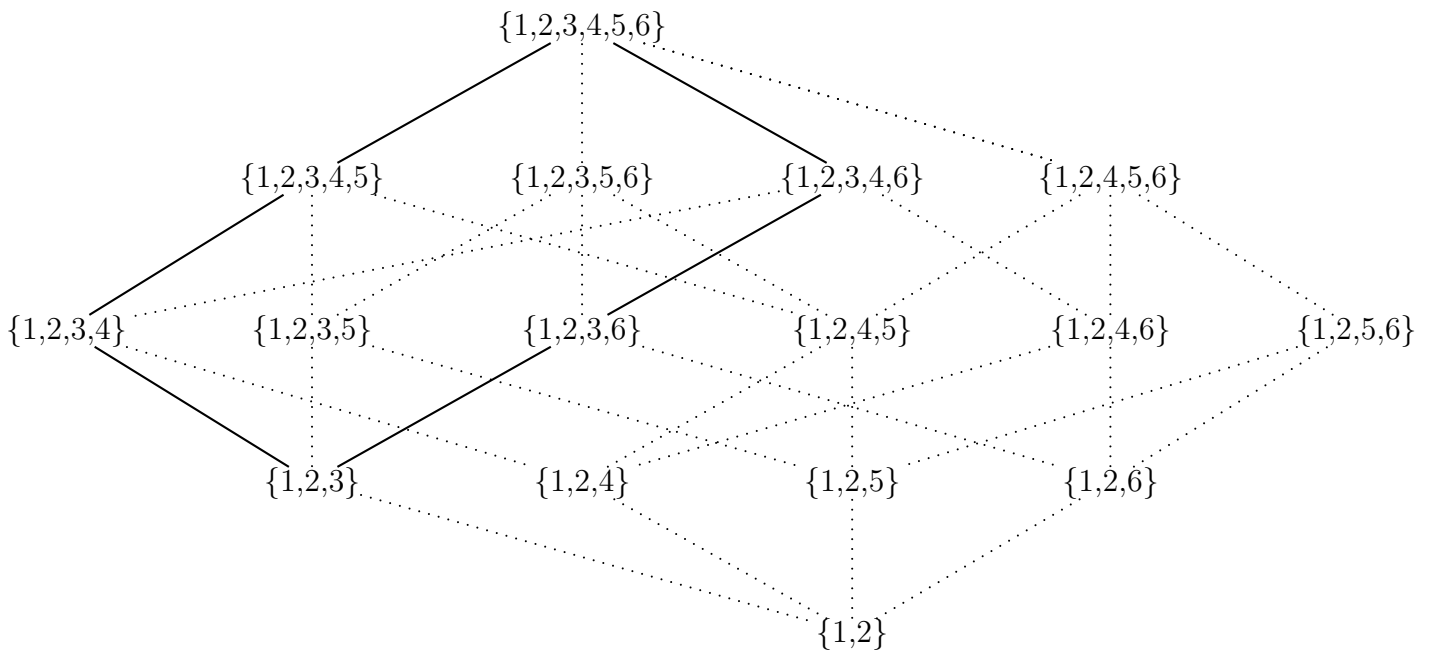
- (Initial Prediction)
- (Subsequent Prediction)
- (Completion)

If successful, then $\langle N, C, R \rangle$ succeeds with $U = U_1 \cup \dots \cup U_k$.

Edge Subsumption

CanBV and ReqBV span sublattices of the powerset lattice of all possible bit vectors.

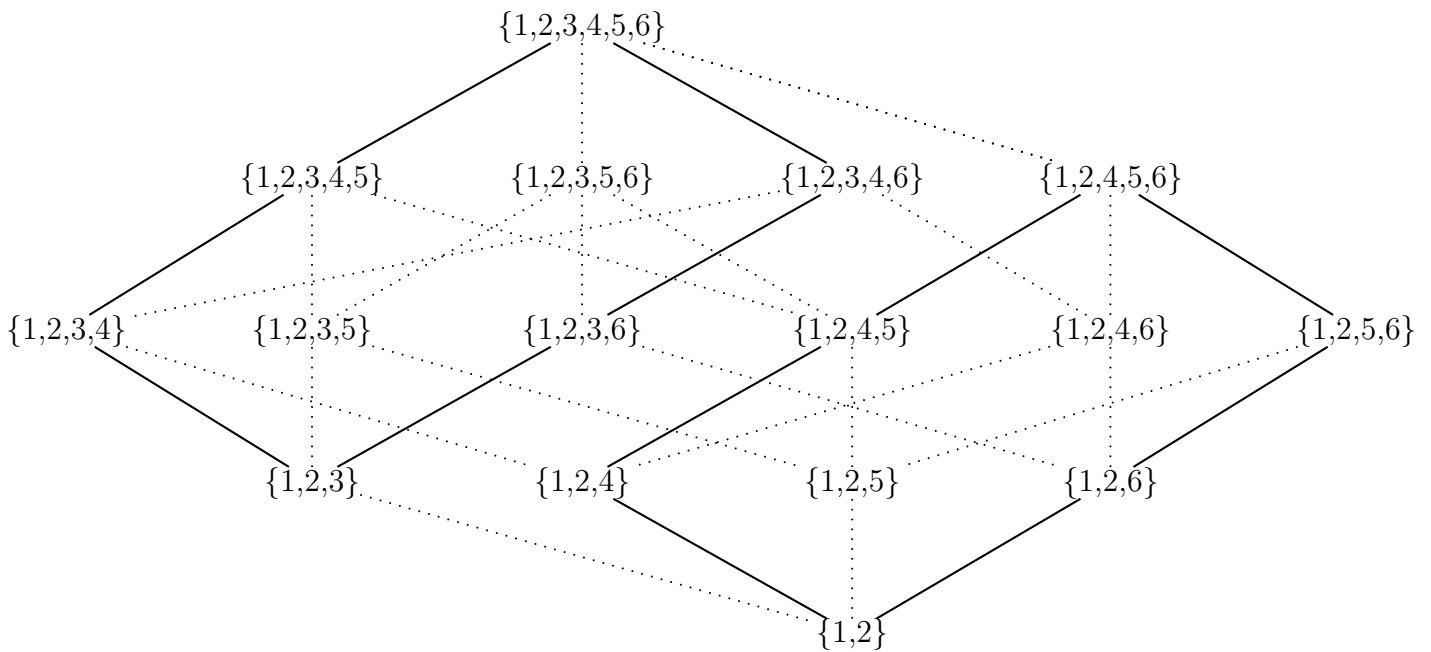
Example: let $C_1 = \{1, 2, 4, 5, 6\}$, $N_1 = N_2$, $C_2 = \{1, 2, 3, 4, 5, 6\}$, and $R_1 = R_2 = \{1, 2\}$:



Edge Subsumption

CanBV and ReqBV span sublattices of the powerset lattice of all possible bit vectors.

Example: let $C_1 = \{1, 2, 4, 5, 6\}$, $N_1 = N_2$, $C_2 = \{1, 2, 3, 4, 5, 6\}$, and $R_1 = R_2 = \{1, 2\}$:



Edge Subsumption

Given existing active chart edge: $\langle N, C_1, R_1 \rangle$

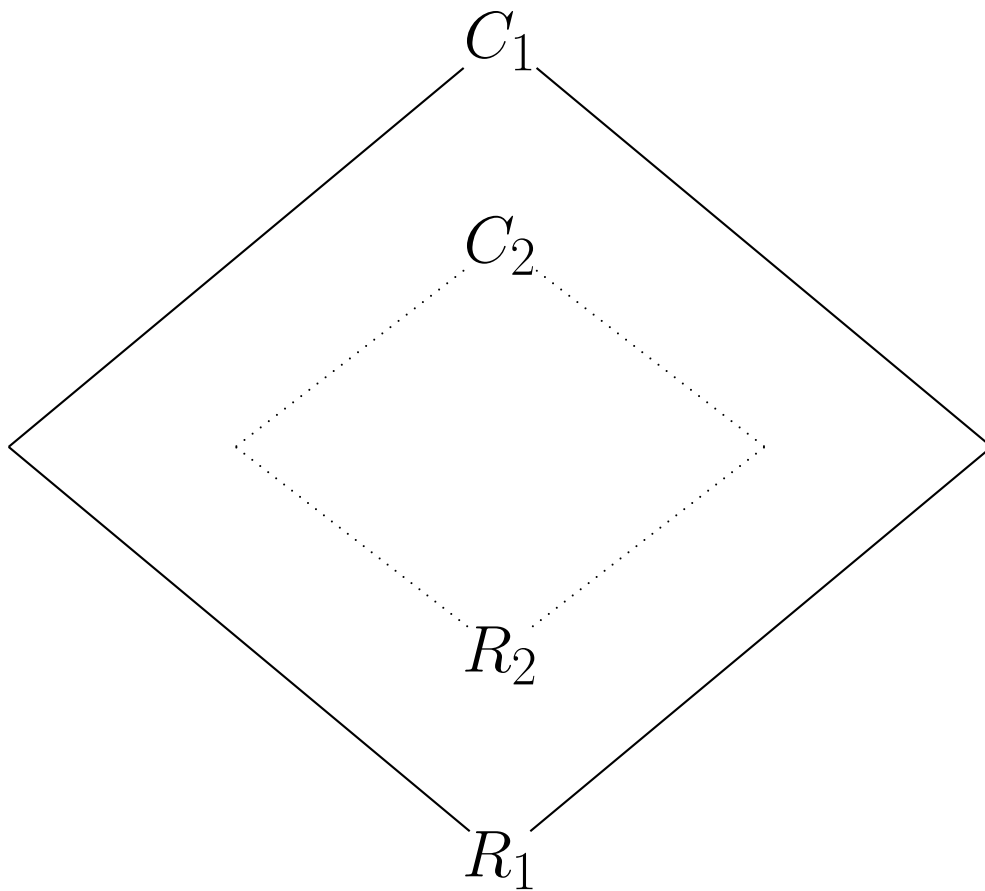
candidate active edge: $\langle N, C_2, R_2 \rangle$

4 main cases: let $O_i := C_i \cap \overline{R_i}$ (*optional*)

1. $R_1 \cap \overline{C_2} \neq \phi$: try another active edge
2. $R_2 \cap \overline{C_1} \neq \phi$: try another active edge
3. $(Z :=) O_2 \cap \overline{C_1} \neq \phi$: then *split* active edge into:
 - (a) $\langle N, C_2, R_2 \cup Z \rangle$ and try another
 - (b) $\langle N, C_2 \cap \overline{Z}, R_2 \rangle$ and try this active edge again.
4. $(Z :=) R_1 \cap O_2 \neq \phi$: try another active edge on $\langle N, C_2 \cap \overline{Z}, R_2 \rangle$,

If none of these, then fail: $\langle N, C_2, R_2 \rangle$ is subsumed by an active edge.

Edge Subsumption



Not 1. $R_1 \cap \overline{C_2} = \phi$

Not 2. $R_2 \cap \overline{C_1} = \phi$

Not 3. $O_2 \cap \overline{C_1} = \phi$

Not 4. $R_1 \cap O_2 = \phi$

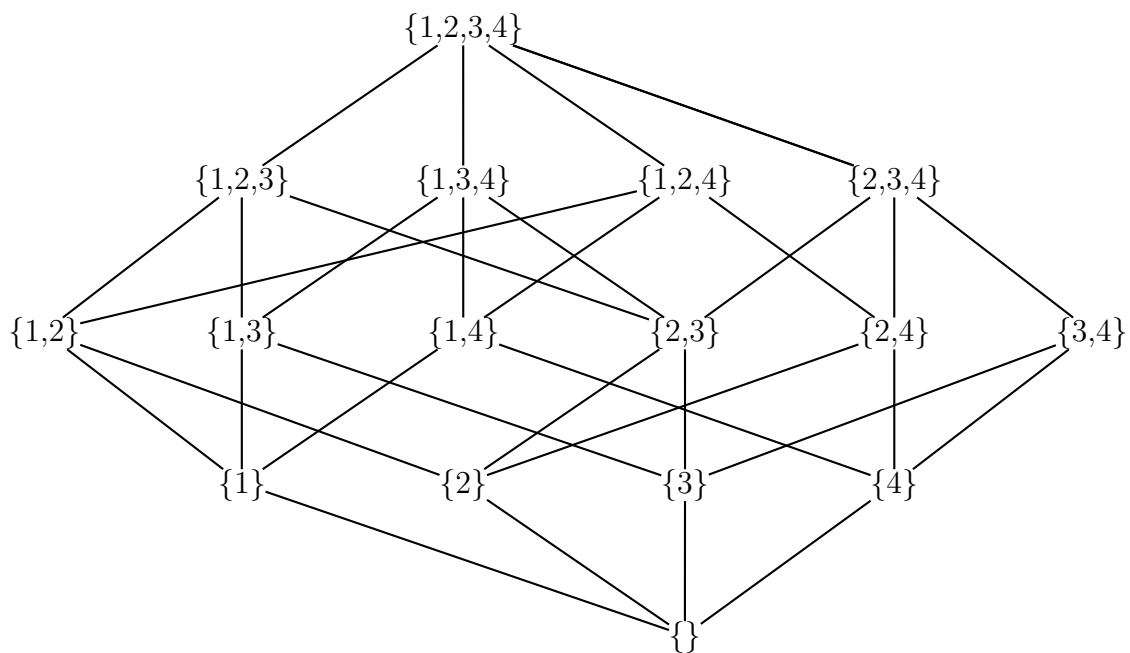
Edge Prediction Revisited

We start with a goal:

$\{1, 2, 3, 4\}$

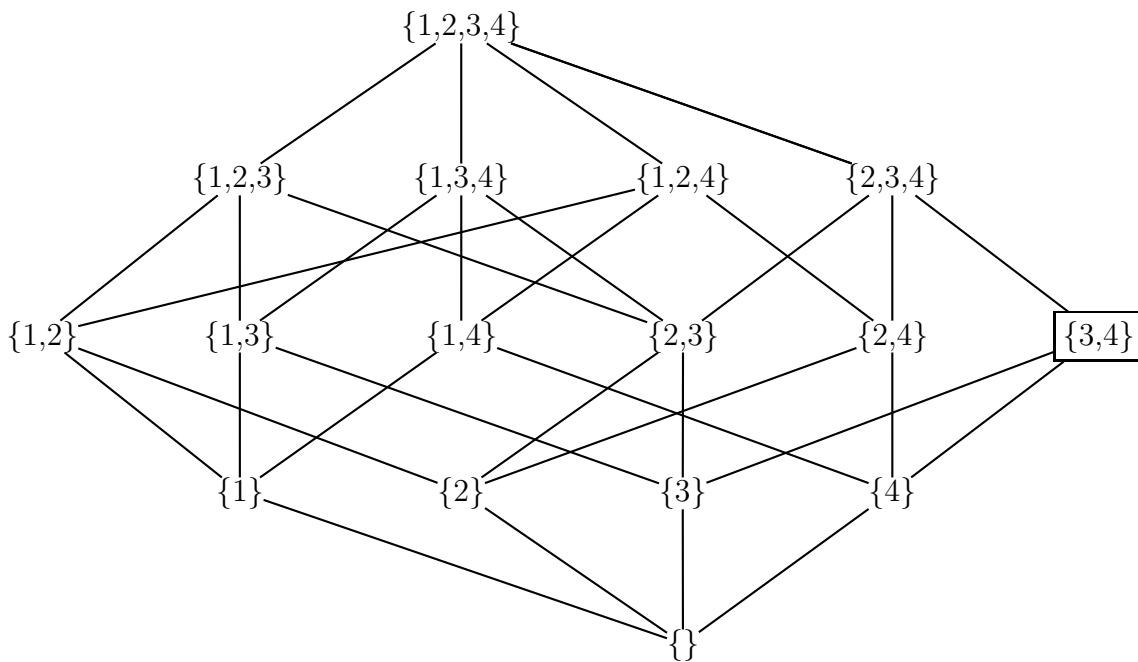
Edge Prediction Revisited

Initial prediction expands it down to its principal ideal (and changes category):



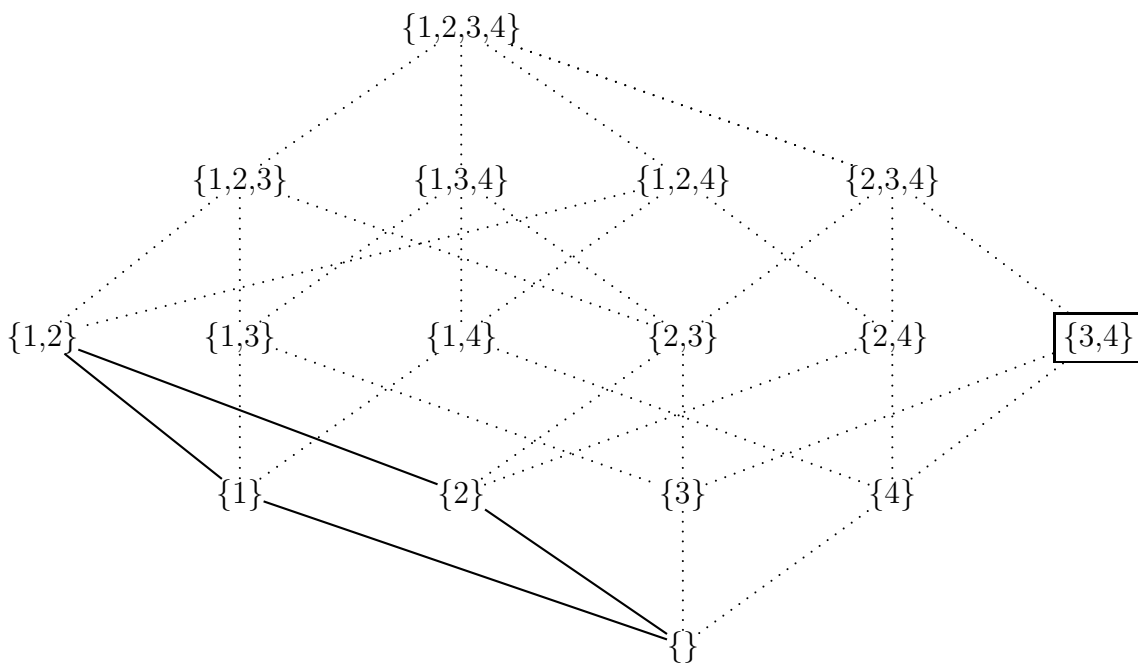
Edge Prediction Revisited

Success identifies a point within that ideal (the UsedBV of the corresponding passive edge):



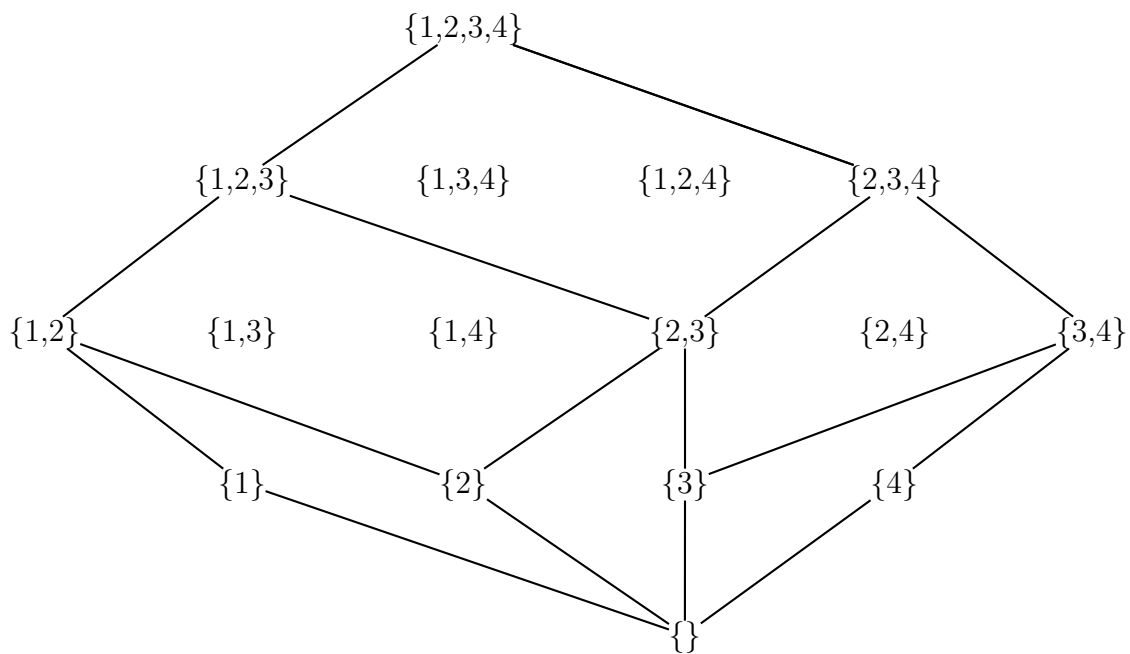
Edge Prediction Revisited

Subsequent prediction computes a pseudo-complement and proceeds with its principal ideal:



Edge Prediction Revisited

LP constraints delete portions of these sublattices as a function of the annotating category:



Category Graphs

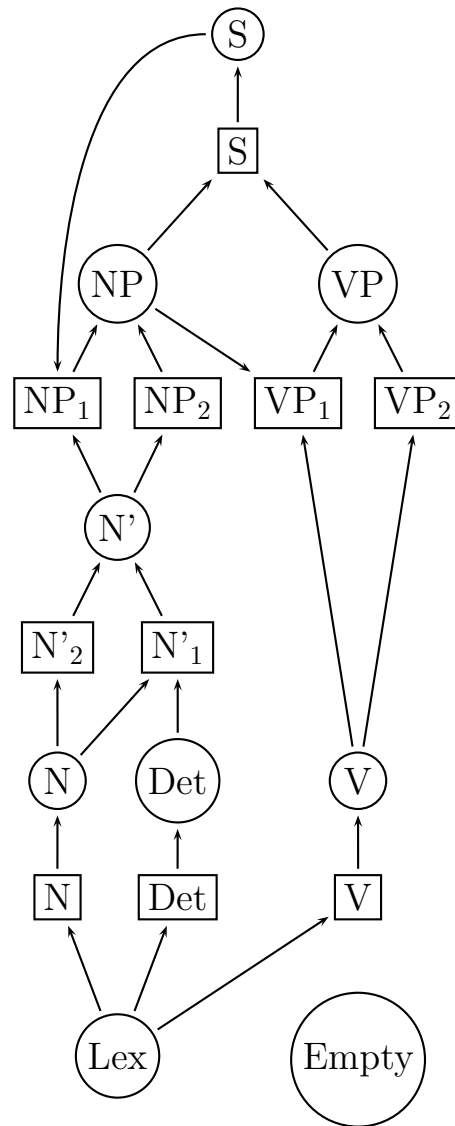
Definition: the *category graph* of grammar G , with ruleset R over non-terminals N , is the smallest directed bipartite graph, $C(G) = \langle V, E \rangle$, such that:

- $V = N \cup R \cup \{\text{Lex}, \text{Empty}\}$,
- $(X, r) \in E$ if non-terminal X appears on the RHS of rule r ,
- $(r, X) \in E$ if the LHS non-terminal of r is X ,
- $(\text{Lex}, r) \in E$ if there is a terminal on the RHS of rule r , and
- $(\text{Empty}, r) \in E$ if r is an empty production rule.

We will call the vertices of $C(G)$ either *category nodes* or *rule nodes*. Lex and Empty are considered category nodes.

Category Graphs

$S \rightarrow VP NP$
 $NP_1 \rightarrow N' S$
 $NP_2 \rightarrow N'$
 $N'_1 \rightarrow N Det$
 $N'_2 \rightarrow N$
 $VP_1 \rightarrow V NP$
 $VP_2 \rightarrow V$
 $N \rightarrow \{\text{boy, girl}\}$
 $Det \rightarrow \{\text{a, the, this}\}$
 $V \rightarrow \{\text{sees, calls}\}$



Yield Bounds

How do we calculate the minimum and maximum yields for the non-terminals of G ?

- Empty has min/max of 0.
- Lex has min/max of 1.
- Topologically sort $C(G)$.
- Compute other min's and max's in topological order.

... unless $C(G)$ has a cycle, in which case every node on a cycle and every node path-accessible from a cycle node has a maximum yield of $+\infty$.

Category Bounds vs. Rule Bounds

This procedure actually assigns max and min yields to rule nodes as well as category nodes.

Bounds on rule nodes are approximations of true category bounds ...

...but they are *more* useful to us, because the bounds they provide are tighter

...and they are only sound when the subtree labelled by their LHS category uses that rule.

How else can we contextualize yield bounds to obtain tighter, more useful numbers?

Height-Dependent Yield Bounds

As a function of tree height, yield bounds are never infinite!

Given non-terminal X , let:

$$X^{max}(\leq h) = \max_{0 \leq j \leq h} X^{max}(j)$$

$$X^{min}(\leq h) = \min_{0 \leq j \leq h} X^{min}(j)$$

Then, for $h > 1$:

$$X^{max}(h) = \max_{X \rightarrow X_1 \dots X_k \in R} \left(\max_{1 \leq i \leq k} X_i^{max}(h-1) + \sum_{j=1, j \neq i}^k X_j^{max}(\leq h-1) \right)$$

$$X^{min}(h) = \min_{X \rightarrow X_1 \dots X_k \in R} \left(\min_{1 \leq i \leq k} X_i^{min}(h-1) + \sum_{j=1, j \neq i}^k X_j^{min}(\leq h-1) \right).$$

Height-Dependent Yield Bounds

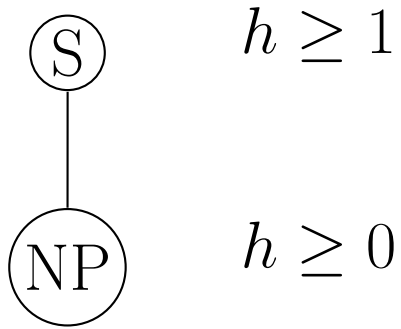
Height-dependent yield bounds are useful because:

- although input in general is unbounded in length, every single input string is of a known and finite length;
- these equations can be iteratively inverted to translate known (bounds on) yield to bounds on height;
- in particular, minimum yield bounds induce maximum height bounds;
- and maximum height bounds restrict depth in a top-down search.

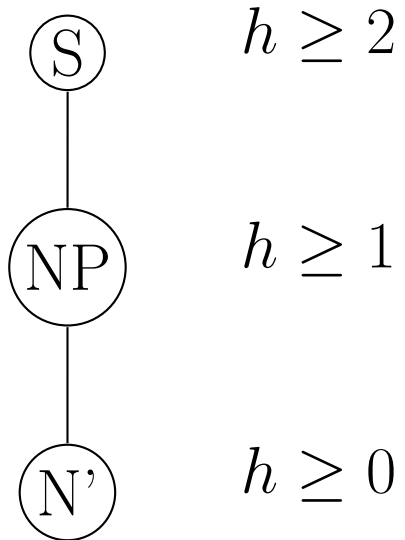
Height-Dependent Yield Bounds

$$\textcircled{S} \quad h \geq 0$$

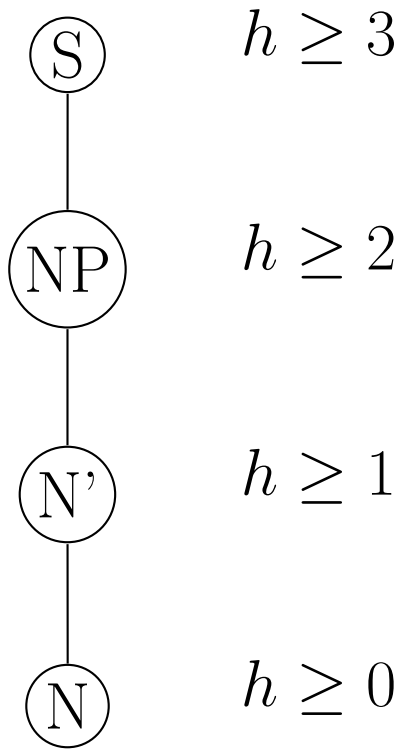
Height-Dependent Yield Bounds



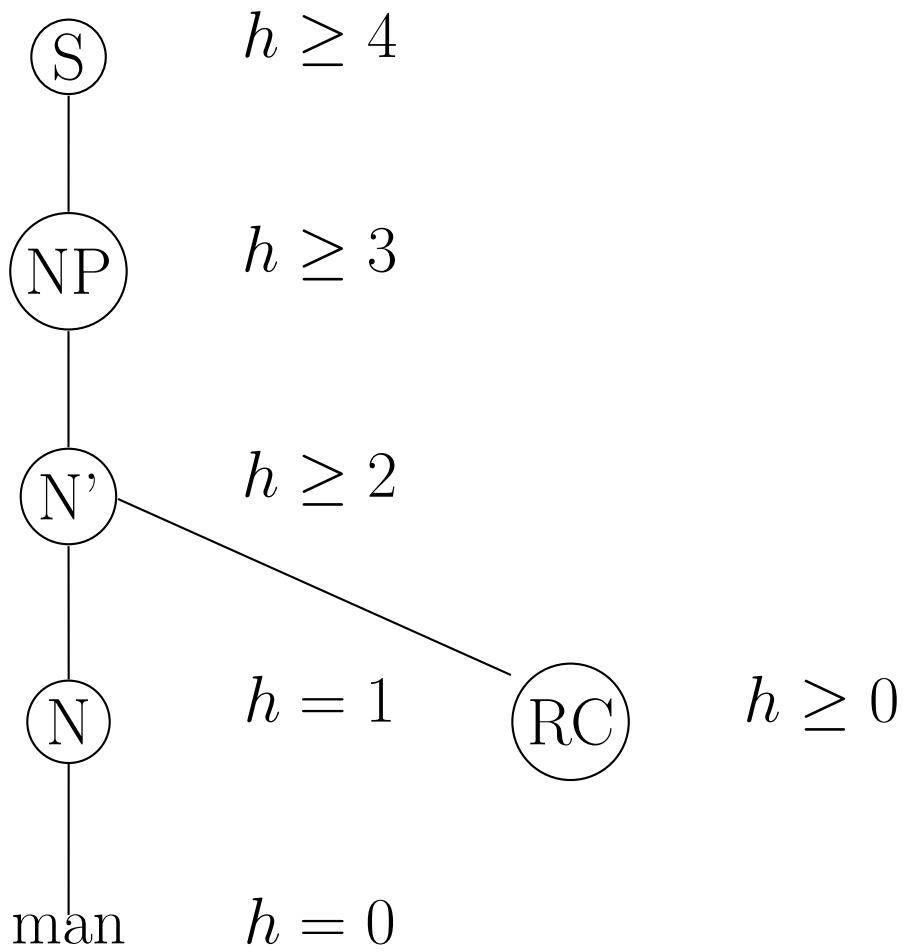
Height-Dependent Yield Bounds



Height-Dependent Yield Bounds



Height-Dependent Yield Bounds



Cycle-Dependent Yield Bounds

It is also possible to bound yields relative to the number of iterations through certain cycles in $C(G)$.

These are more flexible than heights because:

- they can attain exact values without finding a leaf node (lexical item), and
- it is easy to distinguish different cycles or count only some cycles . . .

. . . but there can also be problems with having too many cycle variables, or not knowing which cycles to count.

. . . and I'm running out of time.

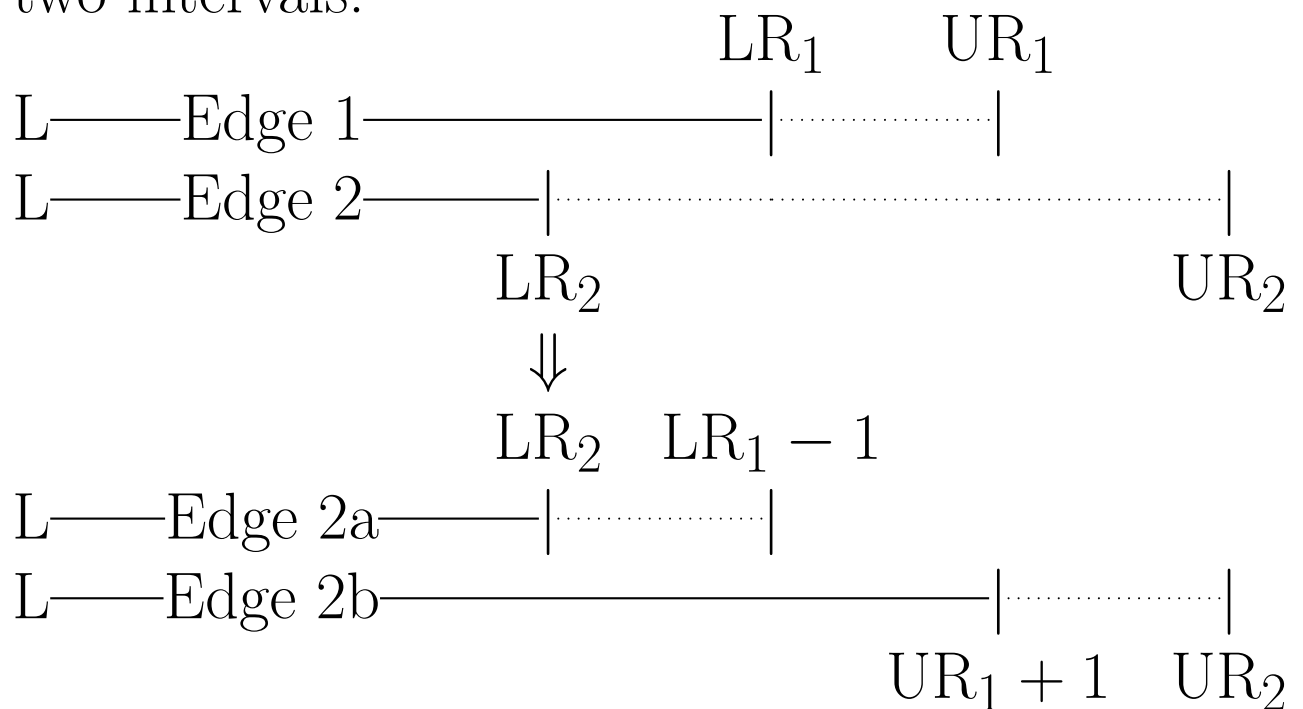
Can CFGs benefit from Yield Bounds?

We checked:

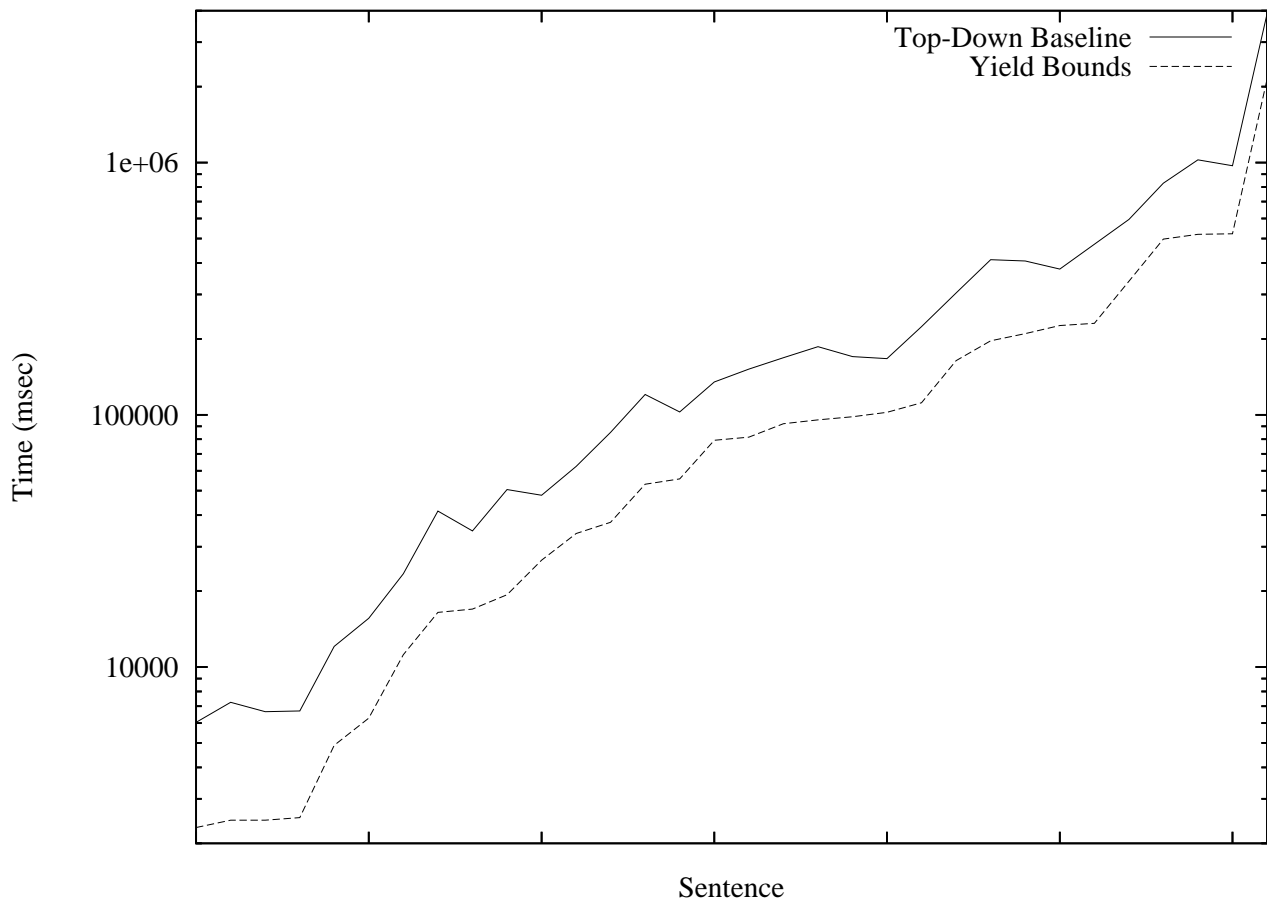
- Grammar induced from WSJ Section 0105 of the Penn Treebank II.
- 2 purely top-down parsers: baseline and with yield bounds.
- Both written in SICStus Prolog.
- Two kinds of active edges in experimental parser:
 - `active(Left,Cat,Right)`
 - `active(Left,Cat,RightLowBound,RightUpBound)`.
- Category bounds and rule bounds used;
 - no height variables,
 - no cycle variables.

Can CFGs benefit from Yield Bounds?

- Prediction and completion both constrained by bounds.
- Subsumption can result in *edge splitting* into two intervals:



Can CFGs benefit from Yield Bounds?



- Yes! (this one, at least)
- Median all-paths parse time reduction: 52.24%
- Maximum: 65.93%
- Minimum: 38.71%

Conclusions

- FWO chart parsing works like CFG chart parsing except we consider all possible sublattices of a powerset lattice instead of all possible line subsegments within a line segment.
- LP constraints delete portions of those sublattices from consideration.
- Category Graphs can be used to compute yield bounds.
- Yield bounds can be used to bound search — even with CFGs.

Conclusions

- Yield bounds come in many flavours:
 - category yields (min and max),
 - rule yields,
 - height-dependent yields,
 - cycle-dependent yields (be careful!).
- Dependent yield equations can be (iteratively) inverted to provide sound and complete depth-bounds on (top-down) search.

Acknowledgements

- Nick Pendar (UT Linguistics, now H5 Inc.)
- Stefan Banjevic (UT Math)
- Michael Demko (UT Computer Science)

Thanks!

Yield Bounds

For typed feature structures, these bounds are approximate:

- we do not know that actual paths exist, just because each edge does;
- unification can bind variables/nodes that propagate along paths, creating potential violations.