Adminstrivia

- Did you attempt Homework 5 yet?
  - center-embedded NPs (S)
Recap

Original grammar (left recursive):
   – **not Prolog-safe!**
1. VP -> Verb, NP
2. VP -> VP, PP

• Right recursive implementation:
   – **Prolog-safe!**
   – produces left recursive trees
   – *(you can't tell it's done right recursively)*
1. v2(V,X) --> pp(PP), v2(V,vp(X,PP)).
2. v2(vp(X,PP),X) --> pp(PP).
3. vp(X) --> verb(V), np(NP), v2(X,vp(V,NP)).
4. vp(vp(V,X)) --> verb(V), np(X).
Recap

Original grammar (left recursive):
   – not Prolog-safe!
1. NP-> Det, NN
2. NP-> NP, PP

• Right recursive implementation:
   – Prolog-safe!
   – produces left recursive trees
   – *(you can't tell it's done right recursively)*
1. np(np(D,NN)) --> det(D), nn(NN).
2. np(X) --> det(D), nn(NN), v(X,np(D,NN)).
3. v(V,X) --> pp(PP), v(V,np(X,PP)).
4. v(np(X,PP),X) --> pp(PP).
Recap

From the course webpage: grammar_transformed2.pl
PP attachment complexity

- Example:
  - I saw a boy with a telescope with a limp with no money
- Stanford Parser:

```
(ROOT
  (S
    (NP (PRP I))
    (VP (VBD saw)
      (NP (DT a) (NN boy))
      (PP (IN with)
        (NP
          (NP (DT a) (NN telescope))
          (PP (IN with)
            (NP (DT a) (JJ limp))))))
    (PP (IN with)
      (NP (DT no) (NN money))))
 (. .))
```

What's the Stanford parser's interpretation for PP attachment here?

Next Question:
3 PPs
  - with a telescope
  - with a limp
  - with no money
How many parses should we expect?
PP attachment complexity

• Example:
  – *I saw a boy with a telescope with a limp with no money*

• How many parses should we expect?

• Let's **modify grammar_transformed2.pl** to check this ...
saw a boy a telescope a limp no money
saw
a boy
 a telescope
  a limp
 no money
saw

a boy

a telescope

a limp

no money
saw
a boy
  a telescope
  a limp
no money
saw
a boy
  a telescope
  a limp
no money
saw a telescope a limp no money a boy
saw
a telescope
a limp
no money
a boy
saw
a telescope
a limp
no money
a boy
saw
  a telescope
  a limp
  no money
a boy
saw a telescope
a limp
no money
a boy
saw
no money
a boy
a telescope
a limp
saw a limp no money a boy a telescope
saw a limp no money a boy a telescope
saw
no money
a boy
a telescope
a limp
Why 14 parses?

- Parses for:
  - *I saw a boy with a telescope with a limp with no money*
- Two attachment points VP (*saw a boy*) and NP (*a boy*) for the three PPs (1. *with a telescope*, 2. *with a limp*, 3. *with no money*)
- Four cases:
  - 1. NP (1,2,3) VP
  - 2. NP (1,2) VP (3)
  - 3. NP (1) VP (2,3)
  - 4. NP VP (1,2,3)

Two items to attach: two possibilities

Total number of parses: 5 + 2 + 2 + 5 = 14
General recursive formula for \# parses

- Let $d_k$ = configuration at depth $k$ ($k$ is depth of last PP)

- **Formula:**
  - $d_k \rightarrow d_1 + \ldots + d_k + d_{k+1}$ (k=1,2,3,...)

- **Start:**
  - $d_1$ (case: one PP, only one place to attach it)
  - $d_1 + d_2$ (case: two PPs)
  - $(d_1 + d_2) + (d_1 + d_2 + d_3) = 2d_1 + 2d_2 + 1d_3$
  - $2(d_1 + d_2) + 2(d_1 + d_2 + d_3) + (d_1 + d_2 + d_3 + d_4) = 5d_1 + 5d_2 + 3d_3 + 1d_4$

Depth:

- XP 1 → 2
- XP 1 → 3
- XP 1 → 2

Total for one PP: 1
Total for two PPs: 2
Total for three PPs: 5
Total for four PPs: 14
General recursive formula for # parses

• In principle, the formula allows to extrapolate the number of syntactically valid parses to an arbitrary number of PPs in a row...

perhaps of academic interest only...
General recursive formula for # parses

• Use the formula in conjunction with the possible attachments at the top level:
  – Case: just one PP phrase, e.g. \([_{PP_1} \ with \ a \ telescope]\)
    • NP (1) VP \(1 \ d_1\)
    • NP VP (1) \(1 \ d_1 = 2 \ (\text{total})\)
  – Case: two PP phrases, e.g. \([_{PP_1} \ with \ a \ telescope]\) \([_{PP_2} \ with \ a \ limp]\)
    • NP (1,2) VP \(d_1 + d_2\)
    • NP (1) VP (2) \(1 \ d_1 \times 1 \ d_1\)
    • NP VP (1,2) \(d_1 + d_2 = 5 \ (\text{total})\)
General recursive formula for # parses

– Case: three PP phrases, e.g. \([_{pp1} with a telescope]\) \([_{pp2} with a limp]\) \([_{pp3} with no money]\)
  
  • NP (1,2,3) VP \(2 d_1 + 2 d_2 + d_3\)
  • NP (1,2) VP (3) \((d_1 + d_2) \times 1 d_1\)
  • NP (1) VP (2,3) \(1 d_1 \times (d_1 + d_2)\)
  • NP VP (1,2,3) \(2 d_1 + 2 d_2 + d_3 = 14 \text{ (total)}\)
General recursive formula for \# parses

• Let’s see if our formula correctly predicts the number of parses for four PPs:
  – Case: three PP phrases, e.g. \[ \text{PP}_1 \text{ with a telescope} \]
    \[ \text{PP}_2 \text{ with a limp} \]
    \[ \text{PP}_3 \text{ with no money} \]
    \[ \text{PP}_4 \text{ with a smile} \]
    
    • NP \((1,2,3,4)\) VP \(5 \, d_1 + 5 \, d_2 + 3 \, d_3 + d_4\)
    • NP \((1,2,3)\) VP \((4)\) \((2 \, d_1 + 2 \, d_2 + d_3) \times 1 \, d_1\)
    • NP \((1,2)\) VP \((3,4)\) \((d_1 + d_2) \times (d_1 + d_2)\)
    • NP \((1)\) VP \((2,3,4)\) \(1 \, d_1 \times (2 \, d_1 + 2 \, d_2 + d_3)\)
    • NP VP \((1,2,3,4)\) \(5 \, d_1 + 5 \, d_2 + 3 \, d_3 + d_4 = 42\) (total)
Counting Parses

• To count the number of parses, run the command:
  
  – `findall(Parse,sentence(Parse, [i,saw,a,boy,with,a,telescope,with,a,limp,with,no,money,with,a, smile], []),List), length(List,Number)).`

  – **Note:**
    • `findall/3` finds all solutions to the `s/3` query, each time it finds a solution it puts it into a list (called `List` here)
    • `length/2`: we evaluate the length of that `List = Number`

?– `findall(Parse,s(Parse,[i,saw,a,boy,with,a,telescope],[[]],List), length(List,Number)).
List = [s(np(i), vp(vbd(saw), np(np(dt(a), nn(boy))), pp(in(with), np(dt(a), nn(telescope))))), ...), s(np(i), vp(vbd(saw), np(dt(a), nn(boy))), pp(in(with), np(dt(a), nn(telescope))))]
Number = 2.`
Counting Parses

Number of parses vs. number of PPs to attach to NP or VP
Counting Parses

<table>
<thead>
<tr>
<th>number</th>
<th>parses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>42</td>
</tr>
<tr>
<td>5</td>
<td>132</td>
</tr>
<tr>
<td>6</td>
<td>429</td>
</tr>
<tr>
<td>7</td>
<td>1430</td>
</tr>
<tr>
<td>8</td>
<td>4862</td>
</tr>
<tr>
<td>9</td>
<td>16796</td>
</tr>
<tr>
<td>10</td>
<td>58786</td>
</tr>
<tr>
<td>11</td>
<td>208012</td>
</tr>
<tr>
<td>12</td>
<td>742900</td>
</tr>
<tr>
<td>13</td>
<td>2674440</td>
</tr>
</tbody>
</table>
Counting Parses

Number of parses increases exponentially...

Graph: Log(number of parses) vs. number of PPs to attach to NP or VP
Catalan Numbers

• See:

\[ C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)! n!} = \prod_{k=2}^{n} \frac{n+k}{k} \quad \text{for } n \geq 0. \]

The first Catalan numbers for \( n = 0, 1, 2, 3, \ldots \) are

1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, … (sequence A000108 in OEIS).