LING/C SC 581:
Advanced Computational Linguistics

Lecture 7

Today's Topic

- *Leaving the topic of context-sensitive languages*
- Turn to writing our own CFGs for natural language:
  1. agreement in natural language
  2. the problem with Prolog & left recursion
  3. a grammar transformation:
     - left recursive to right recursive **BUT** structure preserving
- Homework 4 (*note deadline*)

nl1.prolog

- Example (`nl1.prolog`):

```
1. s --> np, vp.
2. np --> det, nn.
3. det --> [the] | [a].
4. nn --> [man] | [ball].
5. vp --> vtr, np.
6. vtr --> [kicked] | [hit].
```

```
?- [nl1].
true.
?- s([the,man,kicked,the,ball], []).
true ;
false.
?- s([the,man,kicked,the,ball,into,touch], List).
List = [into, touch] ;
false.
```

nl1.prolog

- Enumerate the language:

```
?- s(List, []).
List = [the, man, kicked, the, man] ;
List = [the, man, kicked, the, ball] ;
List = [the, man, kicked, a, man] ;
List = [the, man, kicked, a, ball] ;
List = [the, man, hit, the, man] ;
List = [the, man, hit, the, ball] ;
List = [the, man, hit, a, man] ;
List = [the, man, hit, a, ball] ;
List = [the, ball, kicked, the, man] ;
List = [the, ball, kicked, the, ball] ;
List = [the, ball, kicked, a, man] ;
List = [the, ball, kicked, a, ball] ;
List = [the, ball, hit, the, man] ;
List = [the, ball, hit, the, ball] ;
List = [the, ball, hit, a, man] ;
List = [the, ball, hit, a, ball] ;
List = [a, man, kicked, the, man] ;
List = [a, man, kicked, the, ball] ;
List = [a, man, kicked, a, man] ;
List = [a, man, kicked, a, ball] ;
List = [a, man, hit, the, man] ;
List = [a, man, hit, the, ball] ;
List = [a, man, hit, a, man] ;
List = [a, man, hit, a, ball] ;
List = [a, ball, kicked, the, man] ;
List = [a, ball, kicked, the, ball] ;
List = [a, ball, kicked, a, man] ;
List = [a, ball, kicked, a, ball] ;
List = [a, ball, hit, the, man] ;
List = [a, ball, hit, the, ball] ;
List = [a, ball, hit, a, man] ;
List = [a, ball, hit, a, ball].
```

nl2.prolog

- **Recovering a parse tree**
  - we use a term data structure for the tree
  - simple transformation: adding an extra argument to **all** nonterminals
- Example (`nl2.prolog`):

```
1. s(s(NP, VP)) --> np(NP), vp(VP).
2. np(np(DET, NN)) --> det(DET), nn(NN).
3. det(dt(the)) --> [the].
4. det(dt(a)) --> [a].
5. nn(nn(man)) --> [man].
6. nn(nn(ball)) --> [ball].
7. vp(vp(VTR, NP)) --> vtr(VTR), np(NP).
8. vtr(vbd(kick_ed)) --> [kicked].
9. vtr(vbd(hit_ed)) --> [hit].
```

Basic transformation:

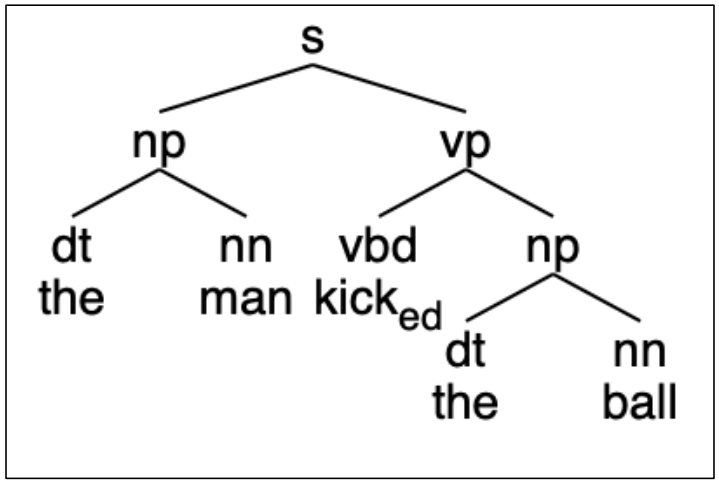
```
x --> y, z.
x(x(Y, Z)) --> y(Y), z(Z).
```

Note: I can return any term I like

nl2.prolog

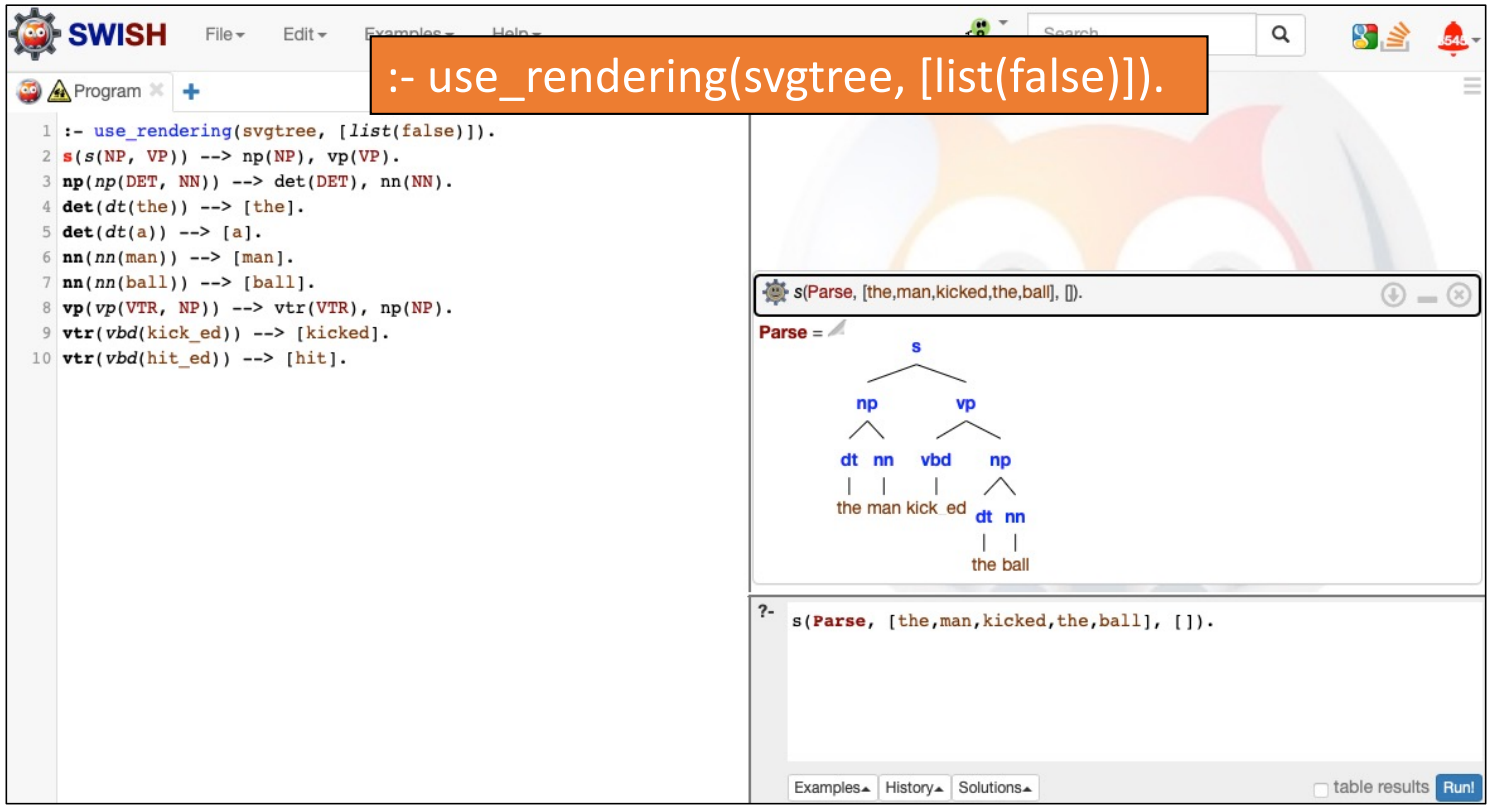
- Example:

```
?- [nl2].
true.
```



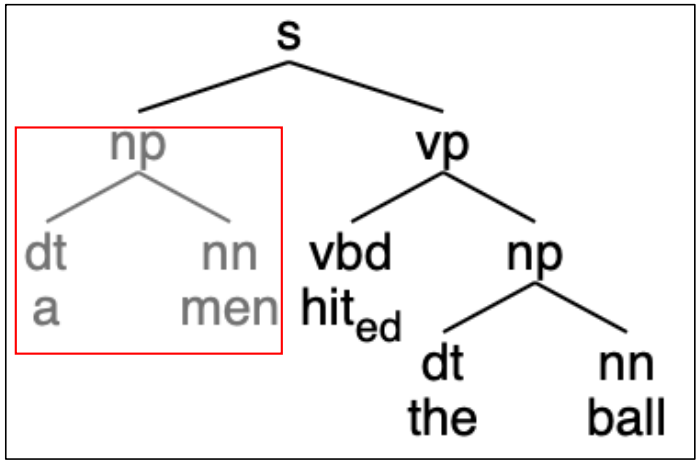

```
?- s(Parse, [the, man, kicked, the, ball], []).
Parse = s(np(dt(the), nn(man)), vp(vbd(kick_ed), np(dt(the),
nn(ball)))).
```

SWISH https://swish.swi-prolog.org



Extra Arguments: Agreement

- **Idea**:
  - We can also use an extra argument to impose constraints between constituents within a DCG rule



- **Example**:
  - English number agreement between DT and NN
  - Data:
    - the man the men
    - a man *a men
  - Lexical Features (Number):
    - *man* value singular (`sg`)
    - *men* value plural (`pl`)
    - *the* value singular or plural (`sg/pl`)
    - *a* value singular (`sg`)

* means ***ungrammatical***

Extra Arguments: Agreement

- Example (`nl3.prolog`):

```
1. s(s(NP, VP)) --> np(NP), vp(VP).
2. np(np(DET, NN)) --> det(DET, NUM), nn(NN, NUM).
3. det(dt(the), sg) --> [the].
4. det(dt(the), pl) --> [the].
5. det(dt(a), sg) --> [a].
6. nn(nn(man), sg) --> [man].
7. nn(nn(men), pl) --> [men].
8. nn(nn(ball), sg) --> [ball].
9. vp(vp(VTR, NP)) --> vtr(VTR), np(NP).
10.vtr(vbd(kick_ed)) --> [kicked].
11.vtr(vbd(hit_ed)) --> [hit].
```

Extra Arguments: Agreement

Note:

- Use of the extra argument NUM for agreement here is basically "syntactic sugar" and **lends no more expressive** power to the grammar rule system
- i.e. *we can enforce agreement without the use of the extra argument at the cost of writing more rules*

- Instead of

```
np(np(DET, NN)) --> det(DET, NUM), nn(NN, NUM).
```

we could have encoded NUM into the nonterminal name:

```
np(np(DET, NN)) --> det_sg(DET), nn_sg(NN).
np(np(DET, NN)) --> det_pl(DET), nn_pl(NN).
det_sg(dt(the)) --> [the].
det_pl(dt(the)) --> [the].
det_sg(dt(a)) --> [a].
nn_sg(nn(man)) --> [man].
nn_pl(nn(men)) --> [men].
nn_sg(nn(ball)) --> [ball].
```

nl4.prolog

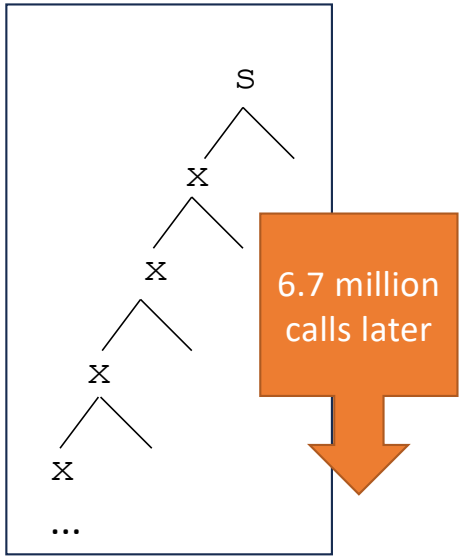
Left recursion and Prolog grammars

Left recursive grammars:

- given Prolog's left-to-right depth-first computation rule, left recursive rules are a *no-no* …
- Example (`left.prolog`):

```
1. s --> x, y.
2. x --> x, [a].
3. x --> [a].
4. y --> [b].
```

rule for nonterminal x immediately calls x again!




```
[?- s([a,b], []).
ERROR: Stack limit (1.0Gb) exceeded
ERROR:   Stack sizes: local: 1.0Gb, global: 35Kb, trail: 1Kb
ERROR:   Stack depth: 6,710,067, last-call: 0%, Choice points: 6,710,059
ERROR:   Probable infinite recursion (cycle):
ERROR:     [6,710,067] user:x([length:2], _9240)
ERROR:     [6,710,066] user:x([length:2], _9266)
   Exception: (6,705,641) x([a, b], _9456) ?
```

Left recursion and Prolog grammars

- Example (`left.prolog`):

```
1. s --> x, y.
2. x --> x, [a].
3. x --> [a].
4. y --> [b].
```

- An idea (*swap rules 2 and 3*):

```
1. s --> x, y.
2. x --> [a].
3. x --> x, [a].
4. y --> [b].
```

; eventually calls for stacking rule 3. *12 million deep*

- (`left2.prolog`)

```
[?- [left2].
true.

[?- s([a,b], []).
true
```

```
[?- s([a,b], []).
true ;
ERROR: Stack limit (1.0Gb) exceeded
ERROR:   Stack sizes: local: 1.0Gb, global: 25Kb, trail: 0Kb
ERROR:   Stack depth: 12,200,438, last-call: 0%, Choice points: 3
ERROR:   Probable infinite recursion (cycle):
ERROR:     [12,200,438] user:x([length:2], _6582)
ERROR:     [12,200,437] user:x([length:2], _6608)
```

Big picture question

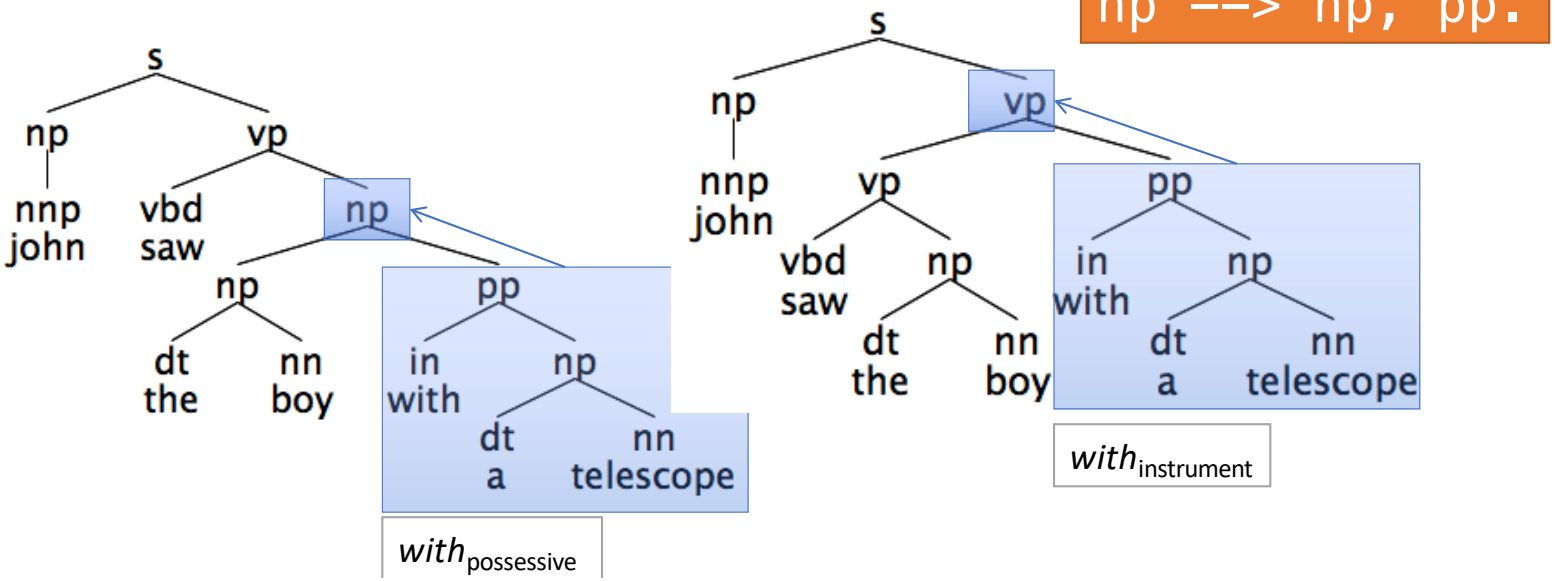
- Is this just a theoretical problem: i.e. not a problem for natural language grammars?
- Unfortunately it is a problem …
  - *John saw the boy with a telescope*
  - is structurally ambiguous wrt. attachment of the PP *with a telescope*
  - (PP = prepositional phrase)

Preposition Phrase (PP) Attachment

- The preferred syntactic analysis is a left recursive parse
- Example:
  - *John saw the boy with a telescope*

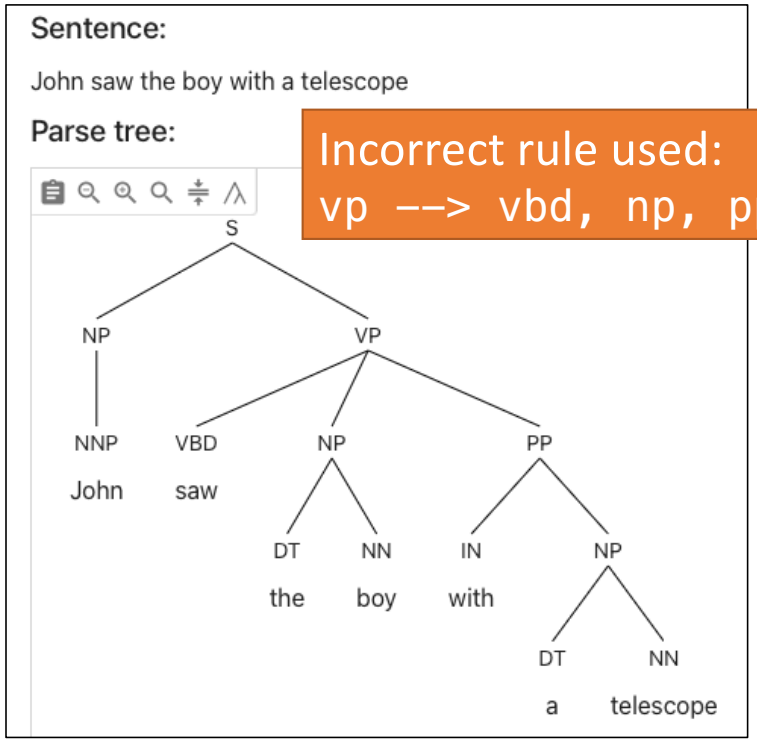
Rules are:

```
vp --> vp, pp.
np --> np, pp.
```



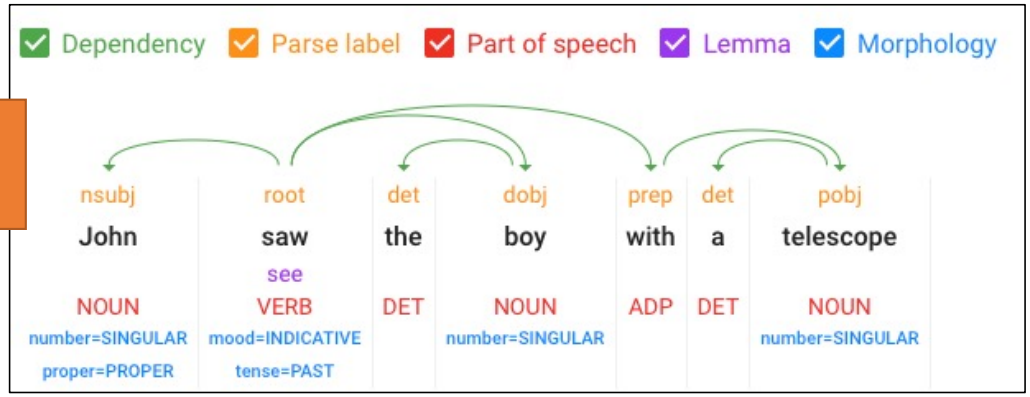
Preposition Phrase (PP) Attachment

https://parser.kitaev.io



Incorrect rule used:
`vp --> vbd, np, pp.`

https://cloud.google.com/natural-language



dependency parse (*essentially same problem*):

```
root is saw
root --> prep
root --> dobj.
```

`nl5.prolog`

Live programming

- Let's add to `nl3.prolog` so we can parse:
  - *John saw the boy with a telescope*
- Need to add:
  - verb (VBD): *saw – past tense* (-ed)
  - preposition (IN): *with*
  - singular nouns (NN): *telescope*, *boy, limp*
  - proper noun (NNP): *john* (*'John'*), *mary – initial caps = variable*
- Need to add:
  - PP attachment to NP and VP rules

Penn Part-of-Speech (POS)Tagset

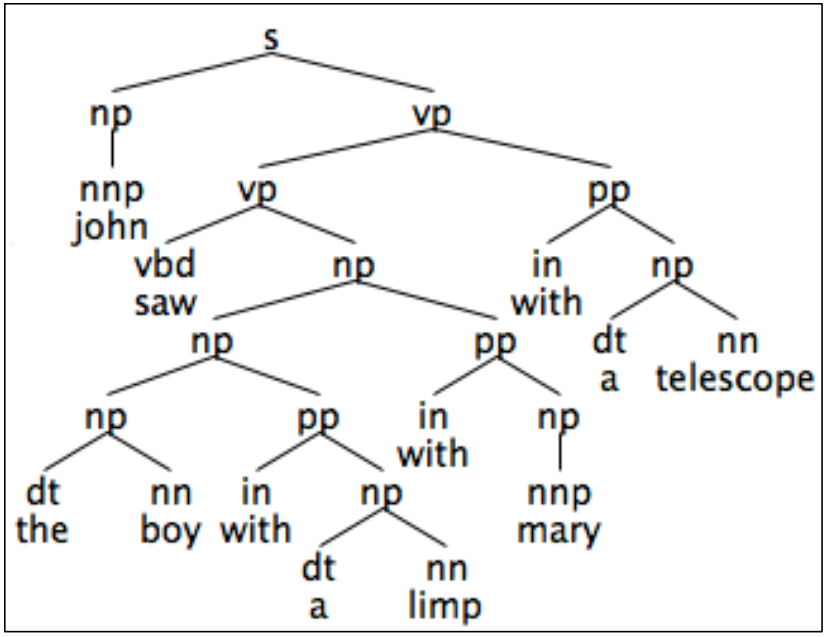
Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past partici-ple	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Figure 8.2 Penn Treebank part-of-speech tags.

Jurafsky & Martin ed3. draft

Preposition Phrase (PP) Attachment

- The preferred syntactic analysis is a left recursive parse
  - notice we can "stack" the PPs, as in:
  - *John saw the boy with a limp with Mary with a telescope*
  - *with*-ambiguity:
    - $with_{possessive}$,
    - $with_{accompaniment}$,
    - $with_{instrument}$



Preposition Phrase Attachment

- Linguistically:
  - PP (recursively) adjoins to NP or VP
  - `np(np(NP,PP)) --> np(NP), pp(PP).`
  - `vp(vp(VP,PP)) --> vp(VP), pp(PP).`
- Left recursion gives Prolog problems
- Derivation (top-down, left-to-right):
  1. vp
  2. vp pp
  3. vp pp pp
  4. vp pp pp pp
  5. vp pp pp pp pp

infinite loop…

Note:
only the parse tree argument shown here
other extra arguments are possible

Transformation

- Apply the general left to right recursive transformation:

```
x(x(X,y)) --> x(X), [y].
x(x(z)) --> [z].
```

```
x(X) --> [z], w(X,x(z)).
x(x(z)) --> [z].
w(W,X) --> [y], w(W,x(X,y)).
w(x(X,y),X) --> [y].
```

Note:
w is a ***fresh*** nonterminal that takes 2 arguments

- to the NP rules:

1. `np(np(DT,NN))` --> `dt(DT,Number)`, [z] `nn(NN,Number).`
2. `np(np(NP,PP))` --> `np(NP)`, `pp(PP)`. [y]

x x

x is the recursive nonterminal

Transformation

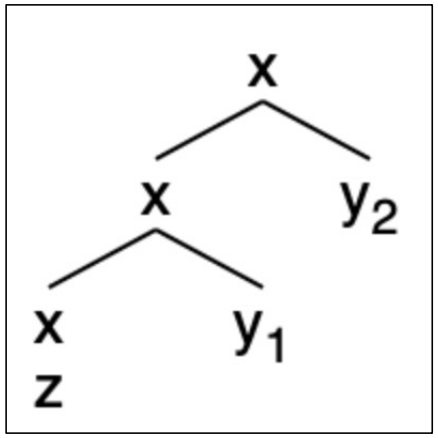
- Consider input strings:
  1. [z]
  2. [z, y]
  3. [z, y_1, y_2]

Parse:
x(z)
x(x(z),y)
x(x(x(z),y_1),y_2)

Transformed rules:
2
1 + 4
1 + 3 + 4

```
1. x(x(X,y)) --> x(X), [y].
2. x(x(z)) --> [z].
```

```
1. x(X) --> [z], w(X,x(z)).
2. x(x(z)) --> [z].
3. w(W,X) --> [y], w(W,x(X,y)).
4. w(x(X,y),X) --> [y].
```



the left recursive structure formed by a right recursive parse of [z, y_1, y_2]

Steps for example 3:

[z, ●y_1, y_2]	rule 1: call nonterminal w(X,x(z))
[z, y_1, ●y_2]	rule 3: call nonterminal w(X,x(x(z),y_1))
[z, y_1, y_2●]	rule 4: answer X = x(x(x(z),y_1),y_2)

Homework 4

- Q1: apply the transformation to the left recursion in `nl5.prolog`:
  - `np(np(NP,PP)) --> np(NP), pp(PP).`
  - `vp(vp(VP,PP)) --> vp(VP), pp(PP).`
- Show your grammar working properly on example sentences:
  1. the boy saw the man with the telescope
  2. the boy with the telescope saw the man
  3. the boy kicked the man with the telescope
  4. the boy with the telescope kicked the man
  5. the boy with the telescope kicked the man with the limp
- Show all possible parses (;) until false in each case
- Q2: suggest a way to limit overgeneration (*no need to implement*)

Homework 4

- Hint #1: consider the case when there are multiple base rules for x
- `x(x(X,y)) --> x(X), [y].`
- `x(x(z)) --> [z].`
- `x(x(w)) --> [w].`
- Hint #2: w must be a fresh nonterminal, i.e. cannot be shared between the NP and VP recursions. Why?
- You can ask questions about the homework in class Tuesday

Homework 4

- Submit to sandiway@arizona.edu
- SUBJECT: 581 Homework 4 *YOUR NAME*
- One PDF file (for grading)
  - include your grammar code and SWI-Prolog screenshots in your answer
- Attach (if I need to run your code):
  - source code for your grammar
- Deadline:
  - midnight Wednesday (*assume HW needs more time*)
  - we will review the homework on Thursday