University of Auckland

07.363

Functional and Logic Programming

Christian Collberg

October 7, 1994

Natural Language II

Proper vs. Common Nouns

Proper noun: Names a person, place or thing. (Jane, Detroit)

Common noun: More general name (girl, city).

- Only common nouns can have a determiner.

```
det --> [a].  det --> [the].
common_noun --> [apple].
proper_noun --> [jill].
proper_noun --> [detroit].
v --> [likes].
s --> np, vp.
np --> proper_noun.
np --> det, common_noun.
vp --> v, np.

?- s([jill, likes, the, apple], []).
   yes
?- s([the, jill, likes, the, apple], []).
   no
```

Slide 15–1

Adjective Phrases I

Adjective: Describes a noun or a pronoun.

Adverb: Modifies an verb, adjective, or adverb. Sometimes **adj + "ly"**.

Brave	*men*	*cry.*		
Adj	N	VP		
John	*likes*	*cute,*	*tall*	*girls.*
NP	V	Adj	Adj	N.
Fergie	*has*	*quite*	*sweet*	*toes.*
NP	V	Adv	Adj	N

Slide 15–2

Adjective Phrases II

```
adj --> [blue].  adj --> [small].
adj --> [pretty].

adv --> [slightly].  adv --> [very].
adv --> [too].  adv --> [pretty].

det --> [the].  det --> [].
n --> [girl].  n --> [i].
v --> [like].

s --> np, vp.
np --> det, ap, n.
ap --> adv, adj, ap.
ap --> adj, ap.
ap --> [].
vp --> v, np.
```

Slide 15–3

Adjective Phrases III

```
?- s([i, like, the, girl], []).
   yes
?- np([the, pretty, girl], []).
   yes
?- np([the, very, pretty, girl], []).
   yes
?- np([the, small, very,
         pretty, girl], []).
   yes
?- np([the, very, small,
         pretty, girl], []).
   yes
?- np([the, pretty, small, girl], []).
   yes
```

Slide 15–4

Subject-Verb Agreement

Subjects have to agree with their verb

1. in number (singular or plural)
2. in person (first, second, third)

Person Agreement:

*	*I*	*loves*	*Bjork*
	1st-pers-sing	3rd-pers-sing	

Number Agreement:

*	*they*	*loves*	*Bjork*
	3rd-pers-plur	3-pers-sing	

Word	Pers	Num	Word	Pers	Num
I	1st	sing	we	1st	plur
you	2nd	sing	you	2nd	plur
he she it	3rd	sing	they	3rd	plur

Slide 15–5

S-V Agreement – Number I

```
det(singular) --> [a].
det(_) --> [the].
det(plural) --> [several].

n(singular) --> [lisa].
n(_) --> [fish].
n(singular) --> [apple].
n(plural) --> [apples].

v(singular) --> [likes].
v(plural) --> [like].

s --> np(Number), vp(Number).
np(Number) --> n(Number).
np(Number) --> det(Number), n(Number).
vp(Number) --> v(Number), np(Number1).
```

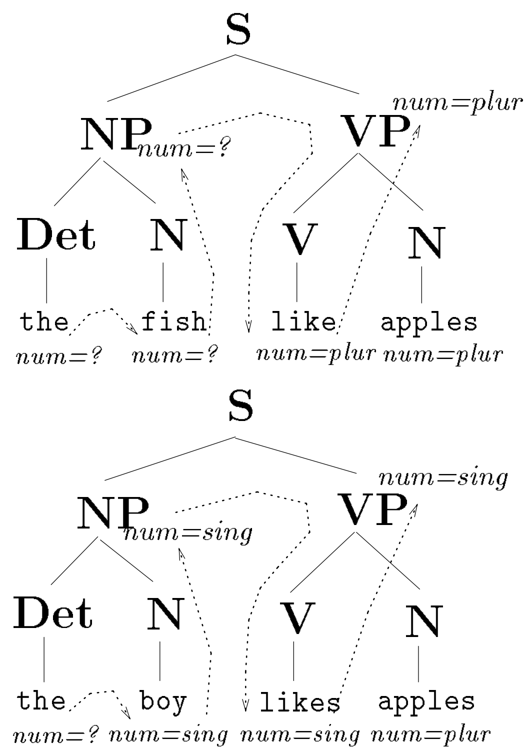
Slide 15–6

S-V Agreement – Number II

```
?- s([lisa, likes, fish], []).
   yes
?- s([lisa, like, fish], []).
   no
?- s([fish, like, apples], []).
   yes
?- s([the, fish, likes, apples], []).
   yes
?- s([the, fish, like, apples], []).
   yes
?- s([a, fish, like, apples], []).
   no
?- s([a, fish, likes, apples], []).
   yes
?-s([several, fish, like, apples],[]).
   yes
```

Slide 15–7

S-V Agreement – Number III



Slide 15–8

S-V Agreement – Person I

```
n(Person) --> [N], {lex_n(N, Person)}.
lex_n(lisa, 3).  lex_n(john, 3).
lex_n(kim, 3).
lex_n(apple, 3).
lex_n(apples, 3).

pronoun(P) --> [N], {lex_pron(N, P)}.
lex_pron(i, 1).  lex_pron(you, 2).
lex_pron(he, 3).  lex_pron(she, 3).
lex_pron(it, 3).  lex_pron(we, 1).
lex_pron(you, 2).  lex_pron(they, 3).

v(Person) --> [V], {lex_v(V, Person)}.
lex_v(likes, 3).  lex_v(like, _).
```

Slide 15–9

S-V Agreement – Person II

```
s --> np(Person), vp(Person).

np(Person) --> n(Person).
np(Person) --> pronoun(Person).
vp(Person) --> v(Person), np(P2).

?- s([lisa, likes, john], []).
   yes
?- s([he, likes, apples], []).
   yes
?- s([i, like, apples], []).
   yes
?- s([you, likes, apples], []).
   no
```

Slide 15–10

Nouns – Possessives I

- Describe something's owner.
- English: NP + **'s**
- Function as determiners.
- Can be compounded.

The boy's	*dog.*
Det	NP

The little boy's	*dog's*	*bone.*
Det	Det	NP

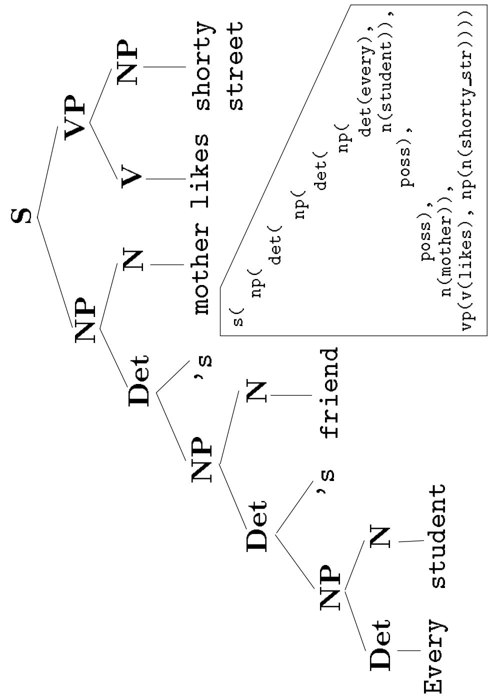
Slide 15–11

Nouns – Possessives II

```
n --> [N], {lex_n(N)}.
lex_n(student).  lex_n(mother).
lex_n(friend).  lex_n(shorty_str).
v --> [likes].

s --> np, vp.
np --> det, n.
np --> n.
vp --> v, np.

det --> [every]
det --> np, ['''', s].

?- s([every, student, '''', s,
   friend, '''', s, mother, likes,
   shorty_str).
   yes
```

Slide 15–12

Nouns – Possessives III



Slide 15–13

Verbs – Subcategorization I

intransitive verbs	take *no* objects
transitive verbs	take *one* objects
bitransitive verbs	take *two* objects
dative verbs	take an NP + a PP

John	*died.*		
NP	*VP*		

John	*likes*	Lisa.	
NP	**VP**	**NP**	

John	*gave*	Lisa	a kiss
NP	**VP**	**NP**	**NP**

John	*gave*	a kiss	to Lisa
NP	**VP**	**NP**	**PP**

Slide 15–14

Verbs – Subcategorization II

```
v(intransitive) --> [died].
v(transitive) --> [likes].
v(ditransitive) --> [gave].
v(dative) --> [gave].

vp --> v(intransitive).
vp --> v(transitive), np.
vp --> v(ditransitive), np, np.
vp --> v(dative), np, pp.
s --> np, vp.
np --> [john].  np --> [lisa].
pp --> [to], np.

?- s([john, likes], []).
   no
?- s([john, likes, lisa], []).
   yes
```

Slide 15–15

Verb Forms I

Main verbs: loves, writes

Auxiliary verbs: could, have, been.

Verb forms: finite, nonfinite, infinitival

FORM	EXAMPLES
finite	halts, halted, writes a program, is halting, has been halting
present participle	halting, write a program
past participle	halted, written a program, been halting
nonfinite	halt, write a program, be halting
infinitival	to have been halting

Slide 15–16

Verb Forms II

- Aux. verbs are attached to a main verb: *have loved.*
- Aux. verbs specify the required *form* of the main verb.
- A sentence must have a *finite* VP.

AUX	RESULT. FORM	REQ. FORM
has	finite	past participle

**Lucy write a book.*
nonfinite

**The program been halting.*
past participle

**Bill writing every program.*
pres part

Slide 15–17

Verb Forms III

```
tv(Form) --> [IV], {lex_tv(IV, Form)}.
lex_tv(write, nonfinite ).
lex_tv(writes, finite ).
lex_tv(wrote, finite ).
lex_tv(written, past_participle ).
lex_tv(writing, present_participle ).

iv(Form) --> [IV], {lex_iv(IV, Form)}.
lex_iv(sleep, nonfinite ).
lex_iv(sleeps, finite ).
lex_iv(slept, finite ).
lex_iv(slept, past_participle ).
lex_iv(sleeping, present_participle ).
lex_iv(go, nonfinite ).
lex_iv(goes, finite ).
lex_iv(went, finite ).
lex_iv(gone, past_participle ).
lex_iv(going, present_participle ).
```

Slide 15–18

Verb Forms IV

```
np --> [lisa].  np --> [book].

aux(Form, Require) --> [Aux],
        {lex_aux(Aux, Form, Require)}.
%     Word     Form        Requires
lex_aux(could, finite, nonfinite).
lex_aux(would, finite, nonfinite).
lex_aux(should, finite, nonfinite).
lex_aux(might, finite, nonfinite).
lex_aux(have, finite, past_participle).
lex_aux(has, finite, past_participle).
lex_aux(been, past_participle,
               present_participle).
lex_aux(be, nonfinite, present_participle).
lex_aux(is, finite, present_participle).
lex_aux(does, finite, nonfinite).
lex_aux(did, finite, nonfinite).
lex_aux(to, infinitival, nonfinite).
```

Slide 15–19

Verb Forms V

```
vp(Form) --> iv(Form).
vp(Form) --> tv(Form), np.
vp(Form) --> aux(Form, Require), vp(Require).
s --> np, vp(finite).

?- s([lisa, could, have, been, sleeping], []).
   yes
?- s([lisa, has, been, sleeping], []).
   yes
?- s([lisa, slept], []).
   yes
?- s([lisa, does, sleep], []).
   yes
?- s([lisa, might, be, sleeping], []).
   yes
?- s([lisa, could, have, is, sleeping], []).
   no
?- s([lisa, has, been, sleep], []).
   no
?- s([lisa, sleeping], []).
   no
?- s([lisa, does, sleeping], []).
```

Slide 15–20

Pronouns – Case Marking I

- Function as a *noun substitute.*
- *Nominative case* is used in the subject position.
- *Accusative case* is used in the object position.
- Possessive articles are determiners.
- Possessive pronouns are NPs.

Personal		Possessive	
nominative	accusative	article	pronoun
I	me	my	mine
you	you	your	yours
he	him	his	his
she	her	her	hers
it	it	its	
we	us	our	ours
you	you	your	yours
they	them	their	theirs

Slide 15–21

Pronouns – Case Marking II

Nominative Case:

I like Lucy.
S V O

Accusative Case:

Lucy likes me.
S V O

**Lucy likes I.*

Possessive Article:

John likes my sister.
NP V Det NP

?John likes me sister.

Possessive Pronoun:

John likes mine.
S V O

Slide 15–22

Pronouns – Case Marking III

```
pers_pron(nom) --> [V],
                   lex_pron(V,_,_,_).
pers_pron(acc) --> [V],
                   lex_pron(_,V,_,_).
poss_article --> [V],lex_pron(_,_,V,_).
poss_pron --> [V],lex_pron(_,_,_,V).

% nom, acc, poss article, poss pron.
lex_pron(i, me, my, mine).
lex_pron(you, you, your, yours).
lex_pron(he, him, his, his).
lex_pron(she, her, her, hers).
lex_pron(it, it, its, 'NONE').
lex_pron(we, us, our, ours).
lex_pron(you, you, your, yours).
lex_pron(they, them, their, theirs).
```

Slide 15–23

Pronouns – Case Marking IV

```
det --> poss_article.
det --> [a].
det --> [the].
n --> [john].
n --> [lucy].
n --> [sister].
v --> [like].
v --> [likes].

s --> np(nom), vp.
np(_) --> det, n.
np(_) --> n.
np(Case) --> pers_pron(Case).
np(_) --> poss_pron.
vp --> v, np(acc).
```

Slide 15–24

Pronouns – Case Marking V

```
?- s([she, likes, john], []).
   yes
?- s([her, likes, john], []).
   no
?- s([i, like, lucy], []).
   yes
?- s([lucy, likes, me], []).
   yes
?- s([john, likes, my, sister],[]).
   yes
?- s([john, likes, hers], []).
   yes
?- s([john, likes, our], []).
   no
```

Slide 15–25

Prepositional Phrases I

- Modifies or specifies actions or objects.
- Can follow a VP or an NP.
- Consists of a preposition + an NP.
- in, on, after, before, with, under, ...

Time:

she	*left*	**after**	*half an hour*
NP	V	P	NP

Place:

a book	**on**	*the floor*
NP	P	NP

Manner:

wrote a program	**with**	*a computer*
VP	P	NP

Slide 15–26

Prepositional Phrases II

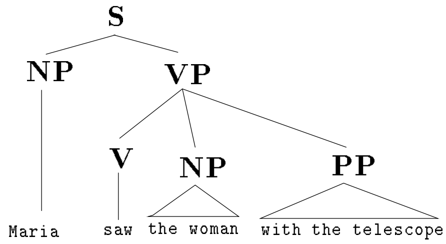
- If a PP can be attached in more than one place we get an *ambiguous* sentence.

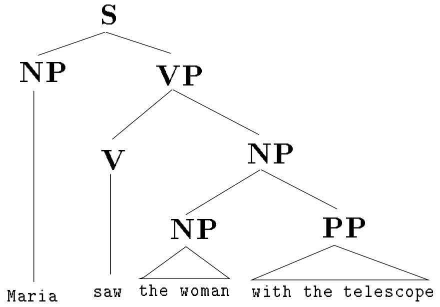
Maria	*saw*	*the woman*	*with the telesc.*
NP	V	NP	PP

```
n(n(telescope))-->[telescope].
n(n(woman))-->[woman].
n(n(maria))-->[maria].  v(v(saw)) --> [saw].
p(p(with)) --> [with].
det(det(the)) --> [the].

s(s(NP,VP)) --> np(NP), vp(VP).
np(np(N, PP)) --> n(N), pp(PP).
np(np(D, N, PP)) --> det(D), n(N), pp(PP).
vp(vp(V, NP, PP)) --> v(V), np(NP), pp(PP).

pp(pp(P, NP)) --> p(P), np(NP).
pp(null) --> [].
```

Slide 15–27

Prepositional Phrases III






```
?- s(S, [maria, saw, the, w, with, the, t], []).
s(np(n(maria)),vp(v(saw),np(det(the),n(woman),
   pp(p(with),np(det(the),n(telescope))))))
s(np(n(maria)),vp(v(saw),np(det(the),n(woman)),
   pp(p(with),np(det(the),n(telescope)))))
```

Slide 15–28

Prepositional Phrases IV

- There can be several consecutive PPs.

Maria	*saw*	*the w*	*in the window*	*with the t*
NP	V	NP	PP	PP

```
np(np(N)) --> n(N).
np(np(Det, N)) --> det(Det), n(N).

np(np(NP, PP)) --> np(NP), pp(PP).

pp(pp(P, NP)) --> p(P), np(NP).


?- s(S, [maria, saw, the, woman,
      in, the, window,
      with, the, telescope], []).
   S = s(np(n(maria)), ....  )  ;
```

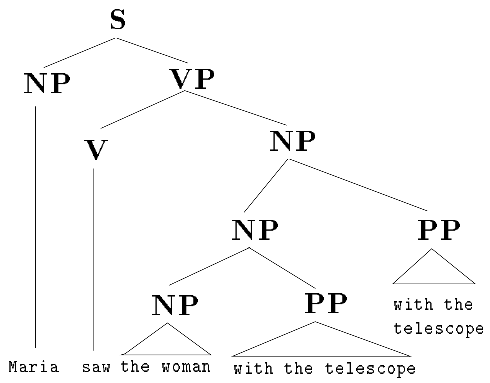
$\boxed{\texttt{loop}}$

Slide 15–29

Prepositional Phrases V

- Left recursive rules can be rewritten to use right recursion

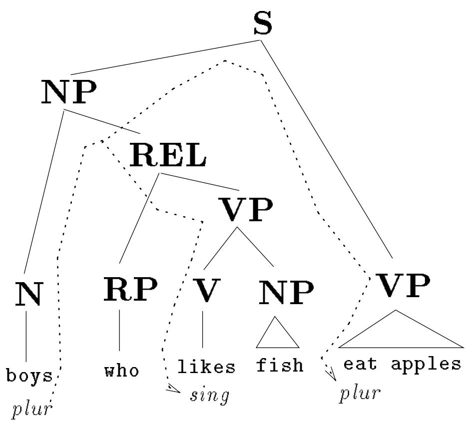
```
np(np(D, N)) --> D(D), n(N).
np(PPs) --> D(D), n(N), pps(np(D, N), PPs).
pp(pp(P, NP)) --> p(P), np(NP).

pps(NP, np(NP, PP)) --> pp(PP).
pps(NP, PPs) --> pp(PP), pps(np(NP, PP), PPs).
```



Slide 15–30

Relative Clauses I

- Modifies or specifies an NP.
- *that, who, whom* + VP.
- Subject-Verb agreement applies.



Slide 15–31

Relative Clauses II

```
det(N/_) --> [Det], {lex_det(Det, N)}.
lex_det(a, singular).
lex_det(the, _).
lex_det(several, plural).

n(N/P) --> [N], {lex_n(N, N, P)}.
lex_n(lisa, singular, 3).
lex_n(boy, singular, 3).
lex_n(boys, plural, 3).
lex_n(fish, _, 3).
lex_n(apple, singular , 3).
lex_n(apples, plural, 3).

v(N/P) --> [V], {lex_v(V, N, P)}.
lex_v(likes, singular, 3).
lex_v(like, _, _).
lex_v(eats, singular, 3).
lex_v(eat, _, _).
```

Slide 15–32

Relative Clauses III

```
rel_pron --> [that].
rel_pron --> [who].  rel_pron --> [whom].

s --> np(Agr), vp(Agr).
np(Agr) --> n(Agr), opt_rel(Agr).
np(Agr) --> det(Agr), n(Agr), opt_rel(Agr).
vp(Agr) --> v(Agr), np(Agr2).

opt_rel(Agr) --> rel_pron, vp(Agr).
opt_rel(Agr) --> [].

?- s([l, who, likes, f, eats, a], []).
   yes
?- s([boys, who, likes, f, eat, a], []).
   no
?- s([boys, who, like, f, eat, a], []).
   yes
```

Slide 15–33

Coordination I

- Conjunctions connect phrases of the same kind.
- *Coordinating conjunctions* (and, or, but) connect Ss, NPs, VPs.
- *Subordinating conjunctions* (if, that, while) connect Ss.

**Lucy loves apples*		*and*	*reads.*
S		Conj	VP
Lucy loves	*apples*	*and*	*oranges.*
	NP	Conj	NP
Lucy loves driving		*if*	*John is quiet.*
S		Conj	SP

Slide 15–34

Coordination II

```
det --> [].  det --> [the].
n --> [john].  n --> [lucy].
n --> [sally].
v --> [eats].  v --> [sleeps].
v --> [loves].  v --> [hates].
v --> [love].

% link same kind of phrases.
coordinating_conj --> [and].
coordinating_conj --> [or].
coordinating_conj --> [but].

% link full clauses only
subordinating_conj --> [if].
subordinating_conj --> [that].
subordinating_conj --> [whether].
subordinating_conj --> [while].
```

Slide 15–35

Coordination III

```
x(s) --> x(np), x(vp).
x(np) --> n.
x(vp) --> v.
x(vp) --> v, x(np).

x(X) -->
   x(X),
   coordinating_conj,
   x(X).

x(s) -->
   x(s),
   subordinating_conj,
   x(s).

?- x(s, [john, eats, or, sleeps], []).
   loop
```

Slide 15–36

Coordination IV

- Rewrite to remove left recursion

```
x(X) --> t(X), r(X).
r(X) --> coordinating_conj, x(X).
r(X) --> [].

x(s) --> t(s), v(s).
v(s) --> subordinating_conj, x(s).
v(s) --> [].

t(s) --> x(np), x(vp).
t(np) --> n.
t(vp) --> v.
t(vp) --> v, x(np).

?- x(s, [john, eats, or, sleeps], []).
   yes % vp or vp
```

Slide 15–37

Coordination V

```
?- x(s, [john, loves, sally, and,
          hates, lucy], []).
   yes % vp and vp
?- x(s, [john, loves, sally, but,
          sally, hates, john], []).
   yes % s and s
?- x(s, [john, and, sally, love, lucy], []).
   yes np and np
?- x(s, [john, loves, sally, while,
          hates, lucy], []).
   no % s while vp
?- x(s, [john, and, sally, and, lucy, ate], []).
   yes % np and np and np
?- x(s, [john, ate], []).
   yes % s
?- x(s, [john, ate, and, sally], []).
   no % s and np
?- x(s, [john, loves, sally, if,
        sally, hates, lucy], []).
   yes % s if s
```

Slide 15–38