LING 364: Introduction to Formal Semantics

Lecture 16

March 7th

Administrivia

- **Homework 2**
  - returned
  - if you didn't get email, let me know
- **Homework 3**
  - should get it back sometime this week

Administrivia

- **New Handout**
  - Chapter 5: Complexities of Referring Expressions
  - read it for next time (Quiz 4)
- **Thursday's Class**
  - in the computer lab (SS 224)
  - exercises based on Chapter 5
  - no homework (*upcoming Spring break*)

Today's Class

- **Two topics**

1. Homework 3 Review
2. Start looking at Chapter 5

Part 1: Homework 3 Review

Homework 3 Review

- **Phrase Structure Grammar (PSG)**
- sbar(**sbar(NP,S)**) --> wh_np(**NP**), s(**S**).
- sbar(**sbar(S)**) --> s(**S**).
- s(**s(VP)**) --> vp(**VP**).
- s(**s(NP,VP)**) --> np(**NP**), vp(**VP**).
- wh_np(**np(who)**) --> [who].
- np(**np(john)**) --> [john].
- np(**np(pete)**) --> [pete].
- np(**np(mary)**) --> [mary].
- np(**np(Det,N)**) --> det(**Det**), n(**N**).
- np(**np(Neg,NP)**) --> neg(**Neg**), np(**NP**).
- np(**np(NP1,Conj,NP2)**) --> np(**NP1**), conj(**Conj**), np(**NP2**).
- neg(**neg(not)**) --> [not].
- conj(**conj(and)**) --> [and].
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- v(**v(is)**) --> [is].
- det(**det(a)**) --> [a].
- n(**n(student)**) --> [student].
- n(**n(baseball_fan)**) --> [baseball,fan].

- **Meaning Grammar (MG)**
- saturate1(**(**P1**,**P2**)**,X) :- **!**, saturate1(P1,X), saturate1(P2,X).
- saturate1(**(\+** P**)**,X) :- **!**, saturate1(P,X).
- saturate1(P,X) :- arg(1,P,X).
- sbar(**P**) --> wh_np(**X**), s(**P**), {saturate1(**P**,**X**)}.
- sbar(**P**) --> s(**P**).
- s(**P**) --> vp(**P**).
- s(**P**) --> np(**X**), vp(**P**), {saturate1(**P**,**X**)}.
- np(**john**) --> [john].
- np(**pete**) --> [pete].
- np(**mary**) --> [mary].
- np(**P**) --> det(**a**), n(**P**).
- np(**(\+ P)**) --> neg, np(**P**).
- np(**(P1,P2)**) --> np(**P1**), conj(**and**), np(**P2**).
- wh_np(**_X**) --> [who].
- neg --> [not].
- conj(**and**) --> [and].
- vp(**P**) --> v(**copula**), np(**P**).
- v(**copula**) --> [is].
- det(**a**) --> [a].
- n(**student(_X)**) --> [student].
- n(**baseball_fan(_X)**) --> [baseball,fan].

Exercise 1

- Modify the PSG to handle
  - (1) Shelby is small
  - (2) Shelby is a dog
  - (3) Hannibal is a dog
- Phrase structure:

  $[_{Sbar}\ [_{S}\ [_{NP}$ Shelby$][_{VP}\ [_{V}$ is$]\ [_{AP}\ [_{A}$ small$]]]]]$
  AP = adjectival phrase, A = adjective

- **NP rules**
- np(**np(john)**) --> [john].
- n(**n(student)**) --> [student].
- add:
- np(np(shelby)) --> [shelby].
- np(np(hannibal)) --> [hannibal].
- n(n(dog)) --> [dog].

- **AP rules**
- ap(ap(A)) --> a(A).
- a(a(small)) --> [small].

- **VP rule**
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- add:
- vp(**vp(V,AP)**) --> v(**V**), ap(**AP**).

Exercise 1

- **The order matters (NP is recursive)**
- sbar(sbar(NP,S)) --> wh_np(NP), s(S).
- sbar(sbar(S)) --> s(S).
- s(s(VP)) --> vp(VP).
- s(s(NP,VP)) --> np(NP), vp(VP).
- wh_np(np(who)) --> [who].
- np(np(john)) --> [john].
- np(np(pete)) --> [pete].
- np(np(mary)) --> [mary].
- **np(np(shelby)) --> [shelby].**
- **np(np(hannibal)) --> [hannibal].**
- np(np(Det,N)) --> det(Det), n(N).
- np(np(Neg,NP)) --> neg(Neg), np(NP).
- np(np(NP1,Conj,NP2)) --> np(NP1), conj(Conj), np(NP2).
- neg(neg(not)) --> [not].
- conj(conj(and)) --> [and].
- **vp(vp(V,AP)) --> v(V), ap(AP).**
- vp(vp(V,NP)) --> v(V), np(NP).
- v(v(is)) --> [is].
- det(det(a)) --> [a].
- n(n(student)) --> [student].
- n(n(baseball_fan)) --> [baseball,fan].
- **n(n(dog)) --> [dog].**
- ap(ap(A)) --> a(A).
- a(a(small)) --> [small].

- **NP rules**
- np(**np(john)**) --> [john].
- n(**n(student)**) --> [student].
- add:
- np(np(shelby)) --> [shelby].
- np(np(hannibal)) --> [hannibal].
- n(n(dog)) --> [dog].

- **AP rules**
- ap(ap(A)) --> a(A).
- a(a(small)) --> [small].

- **VP rule**
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- add:
- vp(**vp(V,AP)**) --> v(**V**), ap(**AP**).

Exercise 1

- Modify the MG to handle
  - (1) Shelby is small
  - (2) Shelby is a dog
  - (3) Hannibal is a dog
- Phrase structure:

  $[_{Sbar}\ [_{S}\ [_{NP}$ Shelby$][_{VP}\ [_{V}$ is$]\ [_{AP}\ [_{A}$ small$]]]]]$

  AP = adjectival phrase, A = adjective

```
 ?- sbar(X,[hannibal,is,a,dog],[]).
X = dog(hannibal) ?
yes
| ?- sbar(X,[shelby,is,small],[]).
X = small(shelby) ?
yes
| ?- sbar(X,[shelby,is,a,dog],[]).
X = dog(shelby) ?
yes
```

- **NP rules**
- np(**john**) --> [john].
- n(**student(_X)**) --> [student].
- add:
- np(shelby) --> [shelby].
- np(hannibal) --> [hannibal].
- n(dog(_X)) --> [dog].

- **AP rules**
- ap(P) --> a(P).
- a(small(_X)) --> [small].

- **VP rule**
- vp(**P**) --> v(**copula**), np(**P**).
- add:
- vp(**P**) --> v(**copula**), ap(**P**).

Exercise 2

- **Possible worlds**
- Part A: using assert with exercise 1 facts
- Part B: Modify MG to handle:
  - (4a) Who is small and a dog?
  - (4b) Who is a dog and not small?
  - **Note**: you need to handle the semantics for "*not small*"

***Note**: looks like we're conjoining dissimilar categories, e.g.*
AP (small) and NP (a dog)
or can view it as a reduced form of sentential conjunction, e.g.
(4a') [$_{sbar}$ Who is small] and [$_{sbar}$ who is a dog]
or that we're conjoining semantically similar types: i.e. predicates with one saturated argument

- **Negation rule**
- np((**\+ P**)) --> neg, np(**P**).
- add
- ap((**\+ P**)) --> neg, ap(**P**).

- **Conjunction rules (simplest way)**
- np((**P1,P2**)) --> np(**P1**), conj(**and**), np(**P2**).
- add:
- np((**P1,P2**)) --> ap(**P1**), conj(**and**), np(**P2**).
- np((**P1,P2**)) --> np(**P1**), conj(**and**), ap(**P2**).

```
?- sbar(X,[who,is,small,and,a,dog],[]).
X = small(_A),dog(_A) ?
| ?- sbar(X,[who,is,a,dog,and,not,small],[]).
X = dog(_A),\+small(_A) ?
```

(**same variable**: enforced from `saturate1/2`)

Exercise 3

- **Relative Clauses**
- Modify PSG and MG to parse:
  - (5) Shelby saw Hannibal
  - (6) Hannibal is who Shelby saw
- Need to handle transitive verbs
- Need to decide on some phrase structure for NP “who Shelby saw”
- e.g.
- $[_{NP}\ [_{NP}$ who$][_{S}\ [_{NP}$ Shelby$][_{VP}\ [_{V}$ saw$]]]]$

- **V rule**
- vp(**vp(V,AP)**) --> v(**V**), ap(**AP**).
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- v(**v(is)**) --> [is].
- add:
- v(**v(saw)**) --> [saw].
- (allows **Shelby saw small* as well)

- **NP rule**
- np(np(WH,S)) --> wh_np(WH), s(S).

- **VP rule (not worrying about empty categories or overgeneration)**
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- add:
- vp(**vp(V)**) --> v(**V**).

Exercise 3

- **Order matters**
- bar(sbar(NP,S)) --> wh_np(NP), s(S).
- sbar(sbar(S)) --> s(S).
- s(s(VP)) --> vp(VP).
- s(s(NP,VP)) --> np(NP), vp(VP).
- wh_np(np(who)) --> [who].
- np(np(john)) --> [john].
- np(np(pete)) --> [pete].
- np(np(mary)) --> [mary].
- np(np(shelby)) --> [shelby].
- np(np(hannibal)) --> [hannibal].
- np(np(Det,N)) --> det(Det), n(N).
- np(np(Neg,NP)) --> neg(Neg), np(NP).
- **np(np(WH,S)) --> wh_np(WH), s(S).**
- np(np(NP1,Conj,NP2)) --> np(NP1), conj(Conj), np(NP2).
- neg(neg(not)) --> [not].
- conj(conj(and)) --> [and].
- vp(vp(V,AP)) --> v(V), ap(AP).
- **vp(vp(V)) --> v(V).**
- vp(vp(V,NP)) --> v(V), np(NP).
- v(v(is)) --> [is].
- **v(v(saw)) --> [saw].**
- det(det(a)) --> [a].
- n(n(student)) --> [student].
- n(n(baseball_fan)) --> [baseball,fan].
- n(n(dog)) --> [dog].
- ap(ap(A)) --> a(A).
- a(a(small)) --> [small].

- **V rule**
- vp(**vp(V,AP)**) --> v(**V**), ap(**AP**).
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- v(**v(is)**) --> [is].
- add:
- v(**v(saw)**) --> [saw].
- (allows **Shelby saw small* as well)

- **NP rule**
- np(np(WH,S)) --> wh_np(WH), s(S).

- **VP rule (not worrying about empty categories or overgeneration)**
- vp(**vp(V,NP)**) --> v(**V**), np(**NP**).
- add:
- vp(**vp(V)**) --> v(**V**).

Exercise 3

- **Relative Clauses**
- Modify MG to parse:
  - (5) Shelby saw Hannibal
  - (6) Hannibal is who Shelby saw
  - we have only `saturate1/2` here

(Lecture 7) Meaning DCG:

```
sentence(P) --> np(NP1), vp(P), {saturate1(P,NP1)}.
vp(P) --> v(P), np(NP2), {saturate2(P,NP2)}.
v(likes(X,Y)) --> [likes].
np(john) --> [john].
np(mary) --> [mary].
saturate1(P,A) :- arg(1,P,A).
saturate2(P,A) :- arg(2,P,A).
```

Transfer new PS rules to MG

- **V rule**
- v(**v(saw)**) --> [saw].
- transfer:
- v(**saw(_X,_Y)**) --> [saw].

- **NP rule (for object relative clause)**
- np(np(WH,S)) --> wh_np(WH), s(S).
- transfer:
- np(P) --> wh_np(WH), s(P), {saturate2(P,WH)}.

- **VP rules**
- vp(**P**) --> v(**copula**), np(**P**).
- add:
- vp(**P**) --> v(P), np(Y), {saturate2(P,Y)}.

- vp(**vp(V)**) --> v(**V**).
- transfer:
- vp(**P**) --> v(**P**).

Exercise 3

- **Relative Clauses**
  - (6) Hannibal is who Shelby saw
- Doesn't work perfectly
- Why?
- **Steps:**
- Hannibal is who Shelby saw
- Hannibal is who Shelby `saw(X,Y)`
- Hannibal is who `saw(shelby,Y)`
- Hannibal is `saw(shelby,Y)`
- Hannibal `saw(shelby,Y)`

s(**P**) --> np(**X**), vp(**P**), {saturate1(**P**,**X**)}.

Transfer new PS rules to MG

- **V rule**
- v(**v(saw)**) --> [saw].
- transfer:
- v(**saw(_X,_Y)**) --> [saw].

- **NP rule (for object relative clause)**
- np(np(WH,S)) --> wh_np(WH), s(S).
- transfer:
- np(P) --> wh_np(WH), s(P), {saturate2(P,WH)}.

- **VP rules**
- vp(**P**) --> v(**copula**), np(**P**).
- add:
- vp(**P**) --> v(P), np(Y), {saturate2(P,Y)}.

- vp(**vp(V)**) --> v(**V**).
- transfer:
- vp(**P**) --> v(**P**).

Exercise 3

- **Relative Clauses**
  - (6) Hannibal is who Shelby saw
- Doesn't work perfectly
- Why?
- **Steps:**
- Hannibal is who Shelby saw
- Hannibal is who Shelby `saw(X,Y)`
- Hannibal is who `saw(shelby,Y)`
- Hannibal is `saw(shelby,Y)`
- Hannibal `saw(shelby,Y)`

s(**P**) --> np(**X**), vp(**P**), {saturate1(**P**,**X**)}.

- **Quick and Dirty Fix**
- s(**P**) --> np(**X**), vp(**P**), {saturate1(**P**,**X**)}.
- s(**P**) --> np(**X**), vp(**P**), {saturate1(**P**,**X**); saturate2(P,X)}.
- **Perhaps a better fix**
- use the lambda calculus
- *idea: semantics of who is λx*
- Hannibal is who `saw(shelby,Y)`
- Hannibal is `lambda(Y,saw(shelby,Y))`
- then saturate/1 works
- **Rule change:**
- np(P) --> wh_np(WH), s(P), {saturate2(P,WH)}.
- adjust to:
- np(lambda(Y,P)) --> wh_np(Y), s(P), {saturate2(P,Y)}.

Exercise 4

- **Adjectives (Intersective interpretation)**
- Modify PSG and MG:
  - (7) Ossie is a bird

    `bird(ossie).`
  - (8) Ossie is tall

    `tall(ossie).`
    - *tall*: predicative adjective
  - (9) Ossie is a tall bird
    - use the **intersective intepretation**
    - i.e. tall(X),bird(X)
    - *a tall bird*
    - $[_{NP}\ [_{Det}$ a$][_{N}\ [_{A}$ tall$][_{N}$ bird$]]]$

- **PSG**
- Step 1:
  - np(np(ossie)) --> [ossie].
  - n(n(bird)) --> [bird].
  - a(a(tall)) --> [tall].
- Step 2:
  - n(n(A,N)) --> a(A), n(N).

- **MG**
  - np(ossie) --> [ossie].
  - n(bird(_X)) --> [bird].
  - a(tall(_X)) --> [tall].
  - n((P1,P2)) --> a(P1), n(P2), {saturate1(P1,X),saturate1(P2,X)}.

Part 2

- Chapter 5: Complexities of Referring Expressions

- (start this today,
- quiz on Thursday in lab class)

Definite NPs

- Semantic differences between:
  - a dog
  - the dog

  - Shelby is a dog
  - Shelby is the dog

Definite NPs

- Definite NP
  - begin with a definite article (“the”)
  - refer or “point” to some entity (in some world)
- Examples
  - the dog
  - the old man
  - the picture of Mary
  - the woman who Susan knows I met

Definite NPs

- Imagine a world
  - Shelby is the only dog which lives at Paul's house
- Then same truth conditions for:
  - (1) Shelby is cute
  - (2) The dog which lives at Paul's house is cute
- Predicates:
  - cute: cute(X). *or* λx.x cute
  - dog: dog(X). *or* λx.x a dog
- What is the role played by "the"?
  - The dog is cute
  - cf. *dog is cute

Definite NPs

- Predicates:
  - cute: cute(X). *or* λx.x cute
  - dog: dog(X). *or* λx.x a dog
- What is the role played by "the"?
  - The dog is cute cf. *dog is cute

- **Idea**:
  - "the" is a function (a **robot** in Chapter 5's terms)
  - takes a property, e.g. dog(X).
  - and picks out something individual in the world, e.g. `dog42`
- **Example**
  - The dog is cute
  - The `dog(X)` is `cute(Y)`
  - `dog42` is `cute(Y)`
  - `cute(dog42).`