LING 364: Introduction to Formal Semantics

Lecture 24

April 13th

Administrivia

- **Homework 4**
  - has been returned
  - if you didn't get an email from me, let me know
- **Homework 5**
  - usual rules
  - due tonight
  - main focus of today's lab session: answer questions you may have

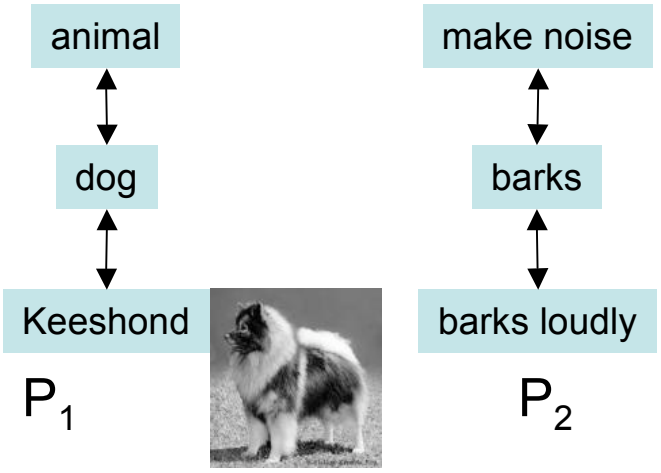
Administrivia

- First, quiz 5 (from last lecture) review...

Table (26) in handout is incorrect ...

Quiz 5

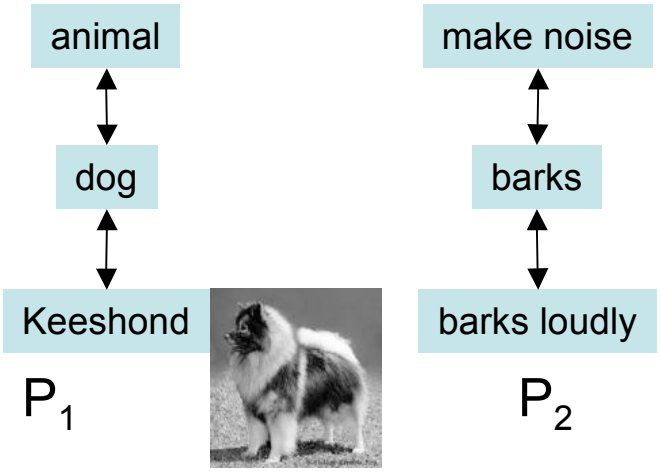
- **Question 1**
  - Is **Some** UE or DE for P_1 and P_2?
    - *some*: $\{X: P_1(X)\} \cap \{Y: P_2(Y)\} \neq \varnothing$
  - Justify your answer using examples of valid/invalid inferences starting from
    - Some dog barks



- **Answer:**
  - **UE for P_1**
  - Some dog barks
  - Some animal barks (OK)
  - Some Keeshond barks
  - **UE for P_2**
  - Some dog barks
  - Some dog makes noise (OK)
  - Some dog barks loudly

Quiz 5

- **Question 2**
  - Is **No** UE or DE for P_1 and P_2?
    - *no*: $\{X: P_1(X)\} \cap \{Y: P_2(Y)\} = \varnothing$
  - Use
    - No dog barks



- **Answer:**
  - **DE for P_1**
  - No dog barks
  - No animal barks
  - No Keeshond barks (OK)
  - **DE for P_2**
  - No dog barks
  - No dog makes noise
  - No dog barks loudly (OK)

Homework 5

- Exercise 1: Truth Tables
- Exercise 2: Universal Quantification and Sets
  - Hint for Question 3
- Exercise 3: Other quantifiers as generalized quantifiers
  - Hint

Exercise 2

- **Assume meaning grammar:**

```
s(M) --> qnp(M), vp(P), {predicate2(M,P)}.
qnp(M) --> q(M), n(P), {predicate1(M,P)}.
q((findall(_X,_P1,L1),findall(_Y,_P2,L2),subset(L1,L
    2))) --> [every].
n(woman(_)) --> [woman].
vp(M) --> v(M), np(X), {saturate2(M,X)}.
v(likes(_X,_Y)) --> [likes].
np(ice_cream) --> [ice,cream].
```

```
saturate1(P,X) :- arg(1,P,X).
saturate2(P,X) :- arg(2,P,X).

subset([],_).
subset([X|L1],L2) :- member(X,L2),
    subset(L1,L2).
member(X,[X|_]).
member(X,[_|L]) :- member(X,L).

predicate1((findall(X,P,_),_),P) :-
    saturate1(P,X).
predicate2((_,(findall(X,P,_),_)),P) :-
    saturate1(P,X).
```

```
?- s(M,[every,woman,likes,ice,cream],[]).
M = findall(_A,woman(_A),_B),findall(_C,likes(_C,ice_cream),_D),subset(_B,_D)
```

Exercise 2

- **Homework Question C (10pts)**
  - Treating names as Generalized Quantifiers (*see below*),
  - Further modify the meaning grammar to handle the sentences
    - Every woman and John likes ice cream
    - John and every woman likes ice cream
  - Evaluate the sentences and submit your runs

Recall Lecture 21

Example

every baby and John likes ice cream

$[_{NP}[_{NP}$ every baby] and $[_{NP}$ John]] likes ice cream

({X: baby(X)} ∪ {X: john(X)}) ⊆ {Y: likes(Y,ice_cream)}

note: set union (∪) is the translation of "*and*"

Define set union as follows:

% L1 ∪ L2 = L3 *"L3 is the union of L1 and L2"*

union(L1,L2,L3) :- append(L1,L2,L3).

Exercise 2

- We can use the rules for *every woman* and rewrite the rules for *John* in the same fashion: i.e. as a **generalized quantifier**:
  - qnp(M) --> q(M), n(P), {predicate1(M,P)}.
  - q((findall(_X,_P1,L1),findall(_Y,_P2,L2),subset(L1,L2))) --> [every].
  - n(woman(_)) --> [woman].
- For example, we can write something like:
  - namenp((findall(X,P,L1),findall(_Y,_P2,L2),subset(L1,L2))) --> name(P), {saturate1(P,X)}.
  - name(john(_)) --> [john].
- Then use this in the grammar:
  - s(M) --> namenp(M), vp(P), {predicate2(M,P)}.
- Query:
  - ?- s(M,[john,likes,ice,cream],[]).
  - M = findall(_A,john(_A),_B),findall(_C,likes(_C,ice_cream),_D),subset(_B,_D)

Exercise 2

- Question becomes how do we merge the following?
  - ?- s(M,[every,woman,likes,ice,cream],[]).
  - M = findall(_A,woman(_A),_B),findall(_C,likes(_C,ice_cream),_D),subset(_B,_D)
  - ?- s(M,[john,likes,ice,cream],[]).
  - M = findall(_A,john(_A),_B),findall(_C,likes(_C,ice_cream),_D),subset(_B,_D)
  - *Notice that predicates 2 and 3 (highlighted in blue) are the same*
- to get:
  - *every woman and John likes ice cream*
  - findall(_A1,woman(_A1),_B1), findall(_A2,john(_A2),_B2), **union(B1,B3,B),**findall(_C,likes(_C,ice_cream),_D),subset(_B,_D)

Exercise 2

- One tool:
  - predicate1((findall(X,P,_),_),P) :- saturate1(P,X).
- can be used to extract the first predicate
- You also have:
  - predicate2((_,(findall(X,P,_),_)),P) :- saturate1(P,X).
- You could easily write a predicate3 rule
  - note: 3rd predicate is not a findall...
- Then you could write a NP conjunction rule like
  - conjnp(((P1a,P1b,union(L1a,L1b,L1)),P2,P3)) --> qnp(M1), [and], namenp(M2), {... *prolog code to pick out and instantiate P1a etc... from M1 and M2*}

Exercise 3

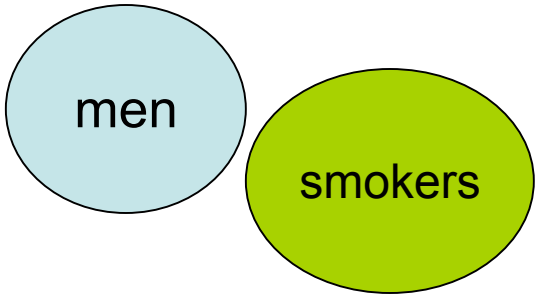
- Other quantifiers can also be expressed using set relations between two predicates:

  Example:

  no: $\{X: P_1(X)\} \cap \{Y: P_2(Y)\} = \varnothing$

  $\cap$ = set intersection

  $\varnothing$ = empty set



  no man smokes

  $\{X: man(X)\} \cap \{Y: smokes(Y)\} = \varnothing$

  should evaluate to true for all possible worlds where there is no overlap between men and smokers

Exercise 3

- How to write set intersection?
  - want to define
  - intersect(L1,L2,L3) such that L3 is L1 ∩ L2
- From lecture 20
  - subset([],_).
  - subset([X|_],L) :- member(X,L).
  - member(X,[X|_]).
  - member(X,[_|L]) :- member(X,L).
- Then using member/2:
  - intersect([],_,[]).
  - intersect([X|L1],L2,L3) :- member(X,L2) -> L3 = [X|L3p], intersect(L1,L2,L3p) ; intersect(L1,L2,L3).