LING 364: Introduction to Formal Semantics

Lecture 18

March 21st

Administrivia

- Welcome back!
- No class this Thursday (I'm out of town)
  - computer lab is reserved for Thursday
  - you are free to use it for the homework
- Homework 4 out today
  - a short homework
  - due next Tuesday (usual rules)
  - email me if you have questions

Administrivia

- Today
  - Quiz 4 Review
  - Continue with Chapter 5
  - Homework 4

Quiz 4 Review

- **Question 1:**
- **Assuming**
  - `s(P) --> name(N), vp(P), {saturate1(P,N)}.`
  - `vp(P) --> v(copula), np_pred(P).`
  - `np_pred(cute(_X)) --> [cute].`
  - `v(copula) --> [is].`
- (1) What would you need to add to make this query work?
  - `?- s(M,[shelby,is,cute],[]).`

1. Answer: `name(shelby) --> [shelby].`

```
?- s(M,[shelby,is,cute],[]).
M = cute(shelby) ?
yes
```

Quiz 4 Review

- **Question 2:**
- Describe in words (or implement)
- What would you need to change to make this query work?
  - ?- s(M,[the,dog,which,lives,at,paul,’\’s’,house,is,cute],[]).

We can already handle the query:
?- np(X,[the,dog,which,lives,at,paul,'\'s',house],[]).
X = dog(_A),lives_at(_A,house(paul))

So we want to compute
dog(X),lives_at(X,house(paul))**,cute(X).**

Quiz 4 Review

```
– np(M) --> [the], n(M).

– np(M) --> name(N), ['''s'], n(M), {saturate1(M,N)}.

– np((M1,M2)) --> np(M1), rel_clause(M2), {saturate1(M1,X), saturate1(M2,X)}.

– n(dog(_X)) --> [dog].
– n(house(_X)) --> [house].

– name(paul) --> [paul].
– name(mary) --> [mary].

– rel_clause(M) --> [which], subj_s(M).

– subj_s(M) --> vp(M).

– vp(M) --> v(M), np(Y), {saturate2(M,Y)}.

– v(lives_at(_X,_Y)) --> [lives,at].

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

from Question 1

```
s(P) --> name(N), vp(P), {saturate1(P,N)}.
vp(P) --> v(copula), np_pred(P).
np_pred(cute(_X)) --> [cute].
v(copula) --> [is].
```

only deals with names

need to add one rule

```
s((P1,P2)) --> np(P1), vp(P2),
{P1=(P3,_), saturate1(P3,X), saturate1(P2,X)}.

?- s(X,[the,dog,which,lives,at,paul,'\'s',house,is,cute],[]).
X = (dog(_A),lives_at(_A,house(paul))),cute(_A)
```

Today’s Topic

- Continue with Chapter 5
- Homework 4

Indefinite NPs

- **(Section 5.3)**
- Contrasting *indefinites* and *definites* with respect to discourse
- **Example**:
  - (6a) A dog came into the house (*followed by*)
  - (6b) The dog wanted some water
- Information-wise:
  - (6a) A dog (*new information*) came into the house
  - (6b) The dog (*old information*) wanted some water
- Novelty-familiarity distinction

Indefinite NPs

- Information-wise:
  - (6a) A dog (*new information*) came into the house
  - (6b) The dog (*old information*) wanted some water
- How to represent this?
- One possibility:
  - (6a) `dog(`**`X`**`), came_into(`**`X`**`,house99).`
  - **Imagine a possible world** (Prolog database):
  - `dog(dog1). dog(dog2). dog(dog3).`
  - `came_into(dog3,house99).`
  - **Query:**
  - `?- dog(X), came_into(X,house99).`
  - `X = dog3`
  - (6b) `wanted(dog3,water).`

Names = concealed descriptions

- **(Section 5.4.1)**
- **Example**:
  - (A) (*Name*) Confucius
  - (B) (*Definite Description*) the most famous Chinese philosopher
- **Similarities**
  - both seem to “pick out” or refer to a single individual
- **One important difference**
  - (B) tells you the criterion for picking out the individual
    - X such that chinese(X), philosopher(X), more_famous_than(X,Y), chinese(Y), philosopher(Y), \+ X=Y.
    - ***is this characterization complete***?
  - (A) doesn’t
  - we trust, in most possible worlds, computation gives us X = confucius

Also saw this earlier for “Shelby” and “the dog which lives at Paul’s house

Names are directly referential

- (Section 5.4.2)
- Kripke: names are non-descriptive
- names refer to things from historical reasons (causal chain)
- **Example** (clear causal history):
  - Baby X is born
  - Parents name it Confucius
  - other people use and accept parent's name
  - gets passed down through history etc...

- (*actually not the best example to use...*)
  - real name: Kong Qiu 孔子
  - styled as "Master Kong" ＝ Confucius: 孔夫子

Names can change their referent

- **(Section 5.4.3)**
- A slight modification from Kripke
- Evans: *social context is important*
- **Example**:
  - Madagascar
    - originally: named part of mainland Africa
    - as a result of Marco Polo's mistake: the island off the coast of Africa

- **Adjectives (Chomsky):**
  - livid as in "livid with rage"
  - pale
  - red

- **Another example (*possibly debunked*):**
  - kangaroo
  - "I don't understand" (aboriginal)
  - *ganjurru* （Guugu Yimidhirr word）

- **Another example:**
  - ono
  - "good to eat" (Hawaiian)

Referential and Attributive Meanings

- **(Section 5.4.4)**
- Russell: definite noun phrases do not refer at all
- Example:
  - the teacher is nice
  - `nice(teacher99).` (*directly referential*)
  - there is exactly one `X` such that `teacher(X), nice(X).`
  - (*attributive: no direct naming*)
- On the attributive reading:
  - the = there is exactly one X such that
  - (i.e. "the" is like a quantifier)

- **Which one is right and does it make any difference?**

Referential and Attributive Meanings

- **(Section 5.4.4)**
- Donnellan: *both are used*
- Example 1:
  - Jones has been charged with Smith's murder
  - Jones is behaving oddly at the trial
  - **Statement**:
  - "Smith's murderer is insane"
  - **referential or attributive use?**

pick out Jones
irrespective of whether he is innocent or not
therefore, referential

- Example 2:
  - everyone loves Smith
  - Smith was brutually murdered
  - **Statement**:
  - "Smith's murderer is insane"
  - **referential or attributive use?**

Smith's murderer = whoever murdered Smith
"quantificational"
therefore, attributive

Plural and Mass Terms

- **(Section 5.5)**
- Godehard Link: Lattice structure
- **horse**:
  - a property, i.e. horse(X) is true for some individuals X given some world (or database)
- **Example**: possible worlds (w1,..,w4)
  - (11) *expressed as a mapping from world to a set of individuals*
    - w1 → {A,B} horse(a). horse(b).
    - w2 → {B,C} horse(b). horse(c).
    - w3 → {A,B,C} horse(a). horse(b). horse(c).
    - w4 → ∅
  - Then
    - meaning of horse in w3 = {A,B,C}
    - meaning of horses in w3 = {A+B,A+C,B+C,A+B+C} (**idea**: *sum*)

Plural and Mass Terms

- Example possible worlds (w1,..,w4):
  - (11) *expressed as a mapping from world to a set of individuals*
    - w1 → {A,B} horse(a). horse(b).
    - w2 → {B,C} horse(b). horse(c).
    - w3 → {A,B,C} horse(a). horse(b). horse(c).
    - w4 → ∅
  - Then
    - meaning of *horse* in w3 = {A,B,C}
    - meaning of *horses* in w3 = {A+B,A+C,B+C,A+B+C} (**idea**: *sum*)
  - In Prolog database form:
    - w3: horse(a). horse(b). horse(c).
    - meaning of *horse*:
      - set of Xs that satisfies the query ?- horse(X).
      - or ?- findall(X,horse(X),List). List = [a,b,c].
    - **meaning of *horses*?**

`findall/3` and `length/2`

- [Introduced previously in lecture 17 slides]
- `findall/3` and `length/2`
  - `findall(X,P,List).`
    - `List` contains each `X` satisfying predicate `P`
  - `length(List,N).`
    - `N` is the length of `List`
- Example:
  - ?- findall(X,dog(X),List), length(List,1).
  - encodes the definite description “the dog”
    - i.e. query holds (i.e. is true) when dog(X) is true and there is a unique X in a given world

Plural and Mass Terms

- **Database (w3):**
  - horse(a).
  - horse(b).
  - horse(c).

  - horses(Sum) :-
  - findall(X,horse(X),L),
  - sum(L,Sum).

  - sum(L,X+Y) :- pick(X,L,Lp), pick(Y,Lp,_).
  - sum(L,X+Sum) :- pick(X,L,Lp), sum(Lp,Sum).

  - pick(X,[X|L],L).
  - pick(X,[_|L],Lp) :- pick(X,L,Lp).

- **Query**:
  - ?- horses(X).
  - X = a+b ? ;
  - X = a+c ? ;
  - X = b+c ? ;
  - X = a+(b+c) ? ;
  - no

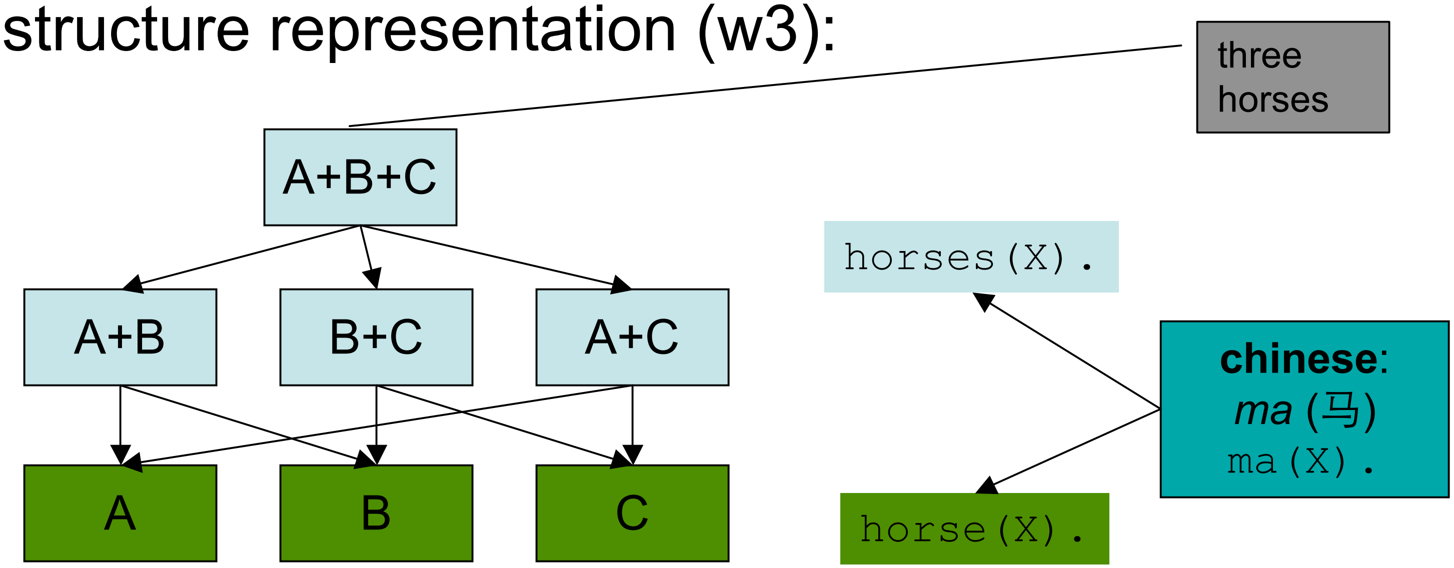
- **Query**:
  - ?- findall(X,horses(X),List).
  - List = [a+b,a+c,b+c,a+(b+c)] ? ;
  - no

Homework 4

- **Question 1 (8pts)**
- (adapted from page 96)
- The proper meaning of *horses* associates a set of plural individuals with each possible world
- Convert the sample meaning for *horse* in w1,..,w4 in (11) into a meaning for *horses*
- **Use Prolog**
  - *for each case, give database and relevant query and output*
- **Question 2 (4pts)**
- Do the same conversion for w5 and w6 below:
  - w5 → {A,B,C,D,E}
  - w6 → {A,B,C,D,E,F}
- **Question 3 (4pts)**
- How would you write the Prolog query for "*three horses*"?
- **Question 4 (4pts)**
- How would you write the Prolog query for "*the three horses*"?

Plural and Mass Terms

- We have:
  - meaning of *horse* in w3 = {A,B,C}
  - meaning of *horses* in w3 = {A+B,A+C,B+C,A+B+C}
- Lattice structure representation (w3):



Plural and Mass Terms

- **Generalizing the lattice viewpoint**
  - *do we have an infinite lattice for mass nouns*?
  - *how do we represent mass nouns*?
- **Mass nouns**: “uncountable”
- **Examples:**
  - gold (*no natural discrete decomposition into countable, or bounded, units*)
  - water
  - furniture *three furnitures
  - three pieces of furniture
  - (unit = one piece)
  - *defines a bounded item which we can count*
- **Compare with:**
  - three horses (English)
  - *does “horses” comes complete with pre-defined units*?
  - three horse-classifier horse (Chinese: sān pǐ mǎ 三匹马)
  - three “units of” horse

Plural and Mass Terms

- One idea:
  - **phrase** **meaning**
  - furniture furniture(X).
  - piece of furniture furniture(X), X is bounded.
  - three pieces of furniture - *requires X to be bounded*
  - | furniture(X) | = 3, X is bounded.
  - *three furniture | furniture(X) | doesn't compute
  - **Chinese: *ma* is like furniture, doesn't come with bounded property**

  - **phrase** **meaning**
  - horses horses(X), X is bounded.
  - three horses | horses(X) | = 3, X is bounded.