LING 364: Introduction to Formal Semantics

Lecture 25

April 18th

Administrivia

- **Homework 5**
  - graded and returned

Administrivia

- Today
  - review homework 5
  - also new handout
    - Chapters 7 and 8
    - we'll begin talking about tense

Homework 5 Review

- **Exercise 1: Truth Tables and Prolog**
- Question A: Using
- ?- implies(P,Q,R1), or(P,Q,R2), \+ R1 = R2.
- for what values of P and Q are P⇒Q and PvQ incompatible?

```
 ?- implies(P,Q,R1), or(P,Q,R2), \+ R1=R2.
P = true,     Q = false,
R1 = false,  R2 = true ? ;
P = false,    Q = false,
R1 = true,   R2 = false ? ;
no
```

P	⇒	Q
T	T	T
F	T	T
F	T	F
T	F	F

R1

P	v	Q
T	T	T
F	T	T
F	F	F
T	T	F

R2

Let P⇒Q = R1
implies(P,Q,**R1**)

PvQ = R2
or(P,Q,**R2**)

compare values

Homework 5 Review

- **Exercise 1: Truth Tables and Prolog**
- **Question B**
- Define truth table and/3 in Prolog

P	^	Q
T	T	T
F	F	T
F	F	F
T	F	F

Result

```
% and(P,Q,Result)
and(true,true,true).
and(true,false,false).
and(false,true,false).
and(false,false,false).
```

Homework 5 Review

- **Exercise 1: Truth Tables and Prolog**
- **Question C**
- Show that
- ¬(P ∨ Q) = ¬P ∧ ¬Q
- (*De Morgan's Rule*)

P	v	Q
T	T	T
F	T	T
F	F	F
T	T	F

P	¬P
T	F
F	T
F	T
T	F

Q	¬Q
T	F
T	F
F	T
F	T

R1

R1	¬R1
T	F
T	F
F	T
T	F

¬ P	∧	¬ Q
F	F	F
T	F	F
T	T	T
F	F	T

R2

```
?- or(P,Q,R1), neg(R1,NR1),
neg(P,NP), neg(Q,NQ),
and(NP,NQ,R2), \+ NR1=R2.

No
```

Homework 5 Review

- **Exercise 1: Truth Tables and Prolog**
- **Question D**
  - Show that
  - $\neg(P \wedge Q) = \neg P \vee \neg Q$
  - (another side of *De Morgan's Rule*)
- Question C was for
- $\neg(P \vee Q) = \neg P \wedge \neg Q$

```
?- and(P,Q,R1),
neg(R1,NR1), neg(P,NP),
neg(Q,NQ), or(NP,NQ,R2),
\+ NR1=R2.
no
```

Homework 5 Review

- **Exercise 2: Universal**
- **Quantification and Sets**
- **Assume meaning grammar:**

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

```
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).
```

every has semantics $\{X: P_1(X)\} \subseteq \{Y: P_2(Y)\}$

every woman likes ice cream {X: woman(X)} ⊆ {Y:likes(Y,ice_cream)}

- ?- s(M,[every,woman,likes,ice,cream],[]).
- M = findall(A,woman(A),B),findall(C,likes(C,ice_cream),D),subset(B,D)

Homework 5 Review

- **Exercise 2: Universal**
- **Quantification and Sets**
- **Questions A and B**
  - John likes ice cream

Simple way
(not using Generalized Quantifiers)

```
s(P) --> namenp(X), vp(P), {saturate1(P,X)}.
namenp(john) --> [john].
```

note: very different from

```
s(M) --> qnp(M), vp(P), {predicate2(M,P)}.
```

```
?- s(M,[john,likes,ice,cream],[]).
M = likes(john,ice_cream)

?- s(M,[john,likes,ice,cream],[]), call(M).
M = likes(john,ice_cream)
```

database

```
woman(mary).
woman(jill).
likes(john,ice_cream).
likes(mary,ice_cream).
likes(jill,ice_cream).
```

Homework 5 Review

- **Exercise 2: Universal Quantification and Sets**
- **Question C**
  - (names as Generalized Quantifiers)
  - Every woman and John likes ice cream
  - ({X: woman(X)} ∪ {X: john(X)}) ⊆ {Y: likes(Y,ice_cream)}
  - John and every woman likes ice cream

Treat *John* just like *every*:

```
s(M) --> namenp(M), vp(P), {predicate2(M,P)}.
namenp((findall(X,P,L1),findall(_Y,_P2,L2),subset(L1,L2))) --> name(P),
{saturate1(P,X)}.
name(john(_)) --> [john].
```

database: john(john)

```
?- s(M,[john,likes,ice,cream],[]).
M = findall(A,john(A),B),findall(C,likes(C,ice_cream),D),subset(B,D))
```

slide is animated

Homework 5 Review

- **Exercise 2: Universal Quantification and Sets**
- **Question C**
  - John and every woman likes ice cream
  - ({X: john(X)} ∪ {Y: woman(Y)}) ⊆ {Z: likes(Z,ice_cream)}

findall **P1** **union** **findall** **P2** **subset** **findall**

Define conjnp:
s(M) --> **conjnp**(M), vp(P), {predicate2(M,P)}.
conjnp(((findall(X,**P1**,L1),findall(Y,**P2**,L2),union(L1,L2,L3)),findall(_,_,L4),subset(L3,L4))) -->
namenp(**M1**), [and], qnp(**M2**), {predicate1(M1,P1), predicate1(M2,P2), saturate1(P1,X), saturate1(P2,Y)}.

```
?- s(M,[john,and,every,woman,likes,ice,cream],[]).
M = (findall(A,john(A),B), findall(C,woman(C),D),union(B,D,E)),
findall(F,likes(F,ice_cream),G),subset(E,G)
```

)

Homework 5 Review

- **Exercise 3: Other Generalized Quantifiers**
- **Question A**

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

  – *No woman likes ice cream*

```
qnp(M) --> q(M), n(P), {predicate1(M,P)}.
q((findall(_X,_P1,L1),findall(_Y,_P2,L2),subset(L1,L2))) --> [every].

q((findall(_X,_P1,L1),findall(_Y,_P2,L2),intersect(L1,L2,[]))) --> [no].
```

```
?- s(M,[no,woman,likes,ice,cream],[]).
M = findall(_A,woman(_A),_B),findall(_C,likes(_C,ice_cream),_D),intersect(_B,_D,[])
?- s(M,[no,woman,likes,ice,cream],[]), call(M).
no
```

Homework 5 Review

- **Exercise 3: Other Generalized Quantifiers**
- **Question A**

  some: $\{X: P_1(X)\} \cap \{Y: P_2(Y)\} \neq \varnothing$

  – *Some women like ice cream* *(plural agreement)*
  – **Some woman likes ice cream*

```
qnp(M) --> q(M), n(P), {predicate1(M,P)}.
q((findall(_X,_P1,L1),findall(_Y,_P2,L2),subset(L1,L2))) --> [every].

q((findall(_X,_P1,L1),findall(_Y,_P2,L2),intersect(L1,L2,L3),\+L3=[])) --> [some].
```

don’t have to implement agreement in this exercise, you could just add:

```
n(woman(_)) --> [women].
v(likes(_X,_Y)) --> [like].
```

Homework 5 Review

- **Exercise 3: Other Generalized Quantifiers**
- **Question A**

  some: $\{X: P_1(X)\} \cap \{Y: P_2(Y)\} \neq \emptyset$

  - *Some women like ice cream* *(plural agreement)*
  - **Some woman likes ice cream*

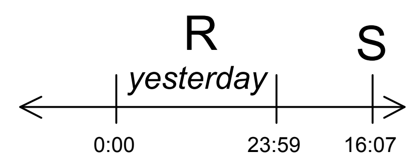
```
qnp(M) --> q(M), n(P), {predicate1(M,P)}.
q((findall(_X,_P1,L1),findall(_Y,_P2,L2),subset(L1,L2))) --> [every].

q((findall(_X,_P1,L1),findall(_Y,_P2,L2),intersect(L1,L2,L3),\+L3=[])) --> [some].
```

```
?- s(M,[some,women,like,ice,cream],[]), call(M).
M =
findall(_A,woman(_A),[mary,jill]),findall(_B,likes(_B,ice_cream),[john,mary,jill]),i
ntersect([mary,jill],[john,mary,jill],[mary,jill]),\+[mary,jill]=[]
```

- Chapter 8: Tense, Aspect and Modality

Tense

- Formal tools for dealing with the semantics of tense (Reichenbach):
  - use the notion of an **event**
  - relate
    - **utterance** or **speech time** (S),
    - **event time** (E) and
    - **reference** (R) *aka* **topic time** (T)
  - S,E and T are **time intervals**:
    - think of them as time lines
    - equivalently, infinite sets containing points of time
  - examples of relations between intervals:
    - precedence (<), inclusion (⊆)



Past Tense

- Example:
  - (16) Last month, I went for a hike
  - S = utterance time
  - E = time of hike
- What can we infer about event and utterance times?
  - E is within the month previous to the month of S
  - (Note: E was completed last month)
- Tense (*went*)
  - past tense is appropriate since E < S

- Reference/Topic time?
  - *may not seem directly applicable here*
  - T = last_month(S)
  - *think of last_month as a function that given utterance time S*
  - *computes a (time) interval*
  - name that interval T

Past Tense

- Example:
  - (16) Last month, I went for a hike
- What can we infer?
  - T = reference or topic time
  - T = last_month(S)
  - $E \subseteq T$
  - *E is a (time) interval, wholly contained within or equal to T*
- Tense (*went*)
  - past tense is appropriate when
  - $T < S$, $E \subseteq T$

Past Tense

- Example:
  - (17) Yesterday, Noah had a rash
- What can we infer?
  - T = yesterday(S)
  - "yesterday" is relative to utterance time (S)
  - E = interval in which Noah is in a state of having a rash
  - E may have begun before T
  - E may extend beyond T
  - E may have been wholly contained within T
  - $E \cap T \neq \varnothing$
- Tense (*had*)
  - appropriate since $T < S$, $E \cap T \neq \varnothing$

expression reminiscent of the corresponding expression for the generalized quantifier *some*

Simple Present Tense

- In English
  - (18a) Mary runs (*simple present*)
  - has more of a **habitual** reading
  - does **not** imply
  - (18b) Mary is running (*present progressive*)
  - T = S, run(mary) true @ T @ T = "at time T"
    - i.e. *Mary is running right now at utterance time*
  - (cf. past: T < S)
- However, the simple present works when we're talking about "states"
- Example: (*has*)
  - (18c) Noah has a rash (*simple present*)
  - rash(noah) true @ T, T=S
  - i.e. Noah has the property of having a rash right now at utterance time

English simple present tense:
T=S, E has a stative interpretation, $E \cap T \neq \emptyset$

Simple Present Tense

- *Some exceptions to the stative interpretation idea*
- Historical Present
  - *present tense used to describe past events*
  - Example:
    - (19a) This guy comes up to me, and he says, “give me your wallet”
    - cf. This guy **came** up to me, and he **said**...
- Real-time Reporting
  - *describe events concurrent with utterance time*
  - Example:
    - (19b) She kicks the ball, and – it’s a goal!
    - cf. She is kicking the ball