LING 364: Introduction to Formal Semantics

Lecture 21

April 4th
Administrivia

• **Homework 3**
  – graded and returned
  – homework 4 should be coming back this week as well
Administrivia

• this Thursday
  – computer lab class
  – fun with quantifiers... homework 5
  – meet in SS 224
Today’s Topic

• Continue with
  – Reading Chapter 6: Quantifiers
  – Quiz 5 (end of class: postponed)
Last Time

- **Quantified NPs:**
  - “*something to do with indicating the quantity of something*”
  - *every* child, nobody
  - *two* dogs, *several* animals
  - *most* people

- **think of quantifiers as “properties-of-properties”**

- every_baby(P) is a proposition

- P: property

- every_baby(P) **true** for P=cried

- every_baby(P) **false** for P=jumped and P=swam

<table>
<thead>
<tr>
<th></th>
<th>every baby</th>
<th>exactly one baby</th>
<th>most babies</th>
</tr>
</thead>
<tbody>
<tr>
<td>cried</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>jumped</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>swam</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

**Generalized quantifiers:**

sets of sets

property = set
Last Time

- **Defining every_baby(P)?**
  - (Montague-style)
  - every_baby(P) is shorthand for
    - \( \lambda P. [\forall X. [\text{baby}(X) \rightarrow P(X)]] \)
    - \( \forall : \text{for all (universal quantifier: logic symbol)} \)

- **Example:**
  - every baby walks
  - [\text{NP every baby}] [\text{vp walks}]
    - \( \lambda P. [\forall X. [\text{baby}(X) \rightarrow P(X)]] \) (walks)
    - \( \forall X. [\text{baby}(X) \rightarrow \text{walks}(X)] \)

- **Prolog-style:**
  - ?- \( \neg \neg (\text{baby}(X), \neg \neg \text{walks}(X)). \) “it’s not true that there is a baby (X) who doesn’t walk”
Conversion to Prolog form

- Show
  - $\forall X. [\text{baby}(X) \rightarrow \text{walks}(X)]$
- is equivalent to (can be translated into):
  - $?- \backslash + (\text{baby}(X), \backslash + \text{walks}(X))$.

We’re going to use the idea that

$\forall X \ P(X)$

is the same as

$\neg \exists X \ \neg P(X)$

let’s call this the “no exception” idea

$\exists = \text{“there exists” (quantifier)}$

(implicitly: all Prolog variables are existentially quantified variables)
Aside: *Truth Tables*

- **Logic of implication**
  - $P \rightarrow Q = \text{(truth value)}$
  - $\begin{array}{c|c|c}
  P & Q & P \rightarrow Q \\
  \hline
  T & T & T \\
  F & T & T \\
  F & F & T \\
  T & F & F \\
  \end{array}$
  - i.e. if $P$ is true, $Q$ must be true in order for $P \rightarrow Q$ to be true
  - if $P$ is false, doesn’t matter what $Q$ is, $P \rightarrow Q$ is true
  - conventionally written as:

$$
\begin{array}{c|c|c|c|c|c|c|c}
\neg P & v & Q & \neg P \vee Q & F & F & T & T \\
\hline
TF & T & T & T & T & T & T & T \\
FT & F & T & F & F & F & F & F \\
TF & T & T & T & T & T & T & T \\
\end{array}
$$

Hence, $P \rightarrow Q$ is equivalent to $\neg (P \vee Q)$
Aside: Truth Tables

- De Morgan’s Rule
- \( \neg(P \lor Q) = \neg P \land \neg Q \)

\[
\begin{array}{c|c|c}
\neg P & \land & \neg Q \\
\hline
\text{FT} & F & \text{FT} \\
\text{TF} & F & \text{FT} \\
\text{TF} & T & \text{TF} \\
\text{FT} & F & \text{FT} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
P & \lor & Q \\
\hline
\text{T} & \text{T} & \text{T} \\
\text{F} & \text{T} & \text{T} \\
\text{F} & \text{F} & \text{F} \\
\text{T} & \text{T} & \text{F} \\
\end{array}
\]

\[
\begin{array}{c|c}
\neg(P \lor Q) \\
\hline
\text{F} \\
\text{F} \\
\text{T} \\
\text{F} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\neg P & \land & \neg Q \\
\hline
\text{F} & \text{F} & \text{T} \\
\text{F} & \text{T} & \text{F} \\
\text{F} & \text{F} & \text{T} \\
\text{F} & \text{T} & \text{F} \\
\end{array}
\]

\[
\begin{array}{c|c}
\neg(P \lor Q) = \text{T} \text{ only when both } P \text{ and } Q \text{ are } \text{F} \\
\neg P \land \neg Q = \text{T} \text{ only when both } P \text{ and } Q \text{ are } \text{F} \\
\end{array}
\]

Hence, \( \neg(P \lor Q) \) is equivalent to \( \neg P \land \neg Q \)
Conversion into Prolog

Note: \(+ (baby(X), \,+walks(X))\) is Prolog for \(\forall X (baby(X) \rightarrow walks(X))\)

Steps:
- \(\forall X (baby(X) \rightarrow walks(X))\)
- \(\forall X (\neg baby(X) \vee walks(X))\)
  - (since \(P \rightarrow Q = \neg P \vee Q\), see truth tables from two slides ago)
- \(\neg \exists X \neg (\neg baby(X) \vee walks(X))\)
  - (since \(\forall X P(X) = \neg \exists X \neg P(X)\), no exception idea from 3 slides ago)
- \(\neg \exists X (baby(X) \land \neg walks(X))\)
  - (by De Morgan’s rule, see truth table from last slide)
- \(\neg (baby(X) \land \neg walks(X))\)
  - (can drop \(\exists X\) since all Prolog variables are basically existentially quantified variables)
- \(+ (baby(X) \land \,+walks(X))\)
  - (\(+ = \) Prolog negation symbol)
- \(+ (baby(X), \,+walks(X))\)
  - (\(, = \) Prolog conjunction symbol)
Last Time

- **Defining every_baby(P)?**
- **(Montague-style)** $\lambda P.[\forall X.\text{baby}(X) \rightarrow P(X)]$
- **(Barwise & Cooper-style)**
- think directly in terms of sets
- leads to another way of expressing the Prolog query

- **Example:** every baby walks
- $\{X: \text{baby}(X)\}$ set of all $X$ such that baby($X$) is true
- $\{X: \text{walks}(X)\}$ set of all $X$ such that walks($X$) is true

- **Subset relation** ($\subseteq$)
- $\{X: \text{baby}(X)\} \subseteq \{X: \text{walks}(X)\}$ the “baby” set must be a subset of the “walks” set

- Imagine a possible world:
  - baby(a).
  - baby(b).
  - baby(c).
  - walks(a).
  - walks(b).
  - walks(c).
  - walks(d).

  - $\{a,b,c\} \subseteq \{a,b,c,d\}$
  - baby $\subseteq$ walks
Subset and Prolog

- How to express this as a Prolog query?
- Findall/3 queries:
  - `?- findall(X,baby(X),L1).`  
    
    `L1` is the set of all babies in the database
  - `?- findall(X,walks(X),L2).`  
    
    `L2` is the set of all individuals who walk

Also need a Prolog definition of the subset relation. For example:

```
subset([],_). % “empty set is a subset of anything”
subset([X|L1],L2) :- member(X,L2), subset(L1,L2).
member(X,[X|_]).
member(X,[_|L]) :- member(X,L).
```

**Prolog Head-Tail List Notation:**

- `[a,b,c]`  
  
  `a` is the head of the list (the first element)
- `[b,c]` is the tail of the list (all but the first element)

we can write a list as follows:

- `[head | tail]`  
  
  `[a | [b,c] ]`

programmatically:

- `[ X | L1]` will match `[a,b,c]`
  when `X = a, L1 = [b,c]`
Generalized Quantifiers

- **Example:** every baby walks
  - \( \{X: \text{baby}(X)\} \subseteq \{X: \text{walks}(X)\} \)
  - the “baby” set must be a subset of the “walks” set
- **Assume the following definitions are part of the database:**
  
  subset([],_).
  subset([X|L1],L2) :- member(X,L2), subset(L1,L2).
  member(X,[X|_ ]).
  member(X,[ _|L]) :- member(X,L).
- **Prolog Query:**
  - \(?- \text{findall}(X, \text{baby}(X), L1), \text{findall}(X, \text{walks}(X), L2), \text{subset}(L1,L2).\)

- **True for world:**
  - baby(a).
  - baby(b).
  - walks(a).
  - walks(b).
  - walks(c).
  - \( L1 = [a,b] \)
  - \( L2 = [a,b,c] \)
  - \(?- \text{subset}(L1,L2) \text{ is true} \)

- **False for world:**
  - baby(a).
  - baby(b).
  - baby(d).
  - walks(a).
  - walks(b).
  - walks(c).
  - \( L1 = [a,b,d] \)
  - \( L2 = [a,b,c] \)
  - \(?- \text{subset}(L1,L2) \text{ is false} \)
Generalized Quantifiers

• **Example**: *every baby walks*
  - (Montague-style) \( \forall X (\text{baby}(X) \rightarrow \text{walks}(X)) \)
  - (Barwise & Cooper-style) \( \{X: \text{baby}(X)\} \subseteq \{X: \text{walks}(X)\} \)

• **how do we define every\_baby(P)?**
  - (Montague-style) \( \lambda P. [\forall X (\text{baby}(X) \rightarrow P(X))] \)
  - (Barwise & Cooper-style) \( \{X: \text{baby}(X)\} \subseteq \{X: P(X)\} \)

• **how do we define every?**
  - (Montague-style) \( \lambda P_1. [\lambda P_2. [\forall X (P_1(X) \rightarrow P_2(X))]] \)
  - (Barwise & Cooper-style) \( \{X: P_1(X)\} \subseteq \{X: P_2(X)\} \)
Quantifiers

• how do we define the expression every?
• (Montague-style) $\lambda P_1.\lambda P_2. [\forall X (P_1(X) \rightarrow P_2(X))]$

Let’s look at computation in the lambda calculus...
• Example: every man likes John
  – **Word**          | **Expression**
  – every            | $\lambda P_1.\lambda P_2. [\forall X (P_1(X) \rightarrow P_2(X))]$
  – man              | man
  – likes            | $\lambda Y.\lambda X. [X \text{ likes } Y]$
  – John             | John
• Syntax: $[S [NP [Q \text{ every}][N \text{ man}]] [VP [V \text{ likes}][NP \text{ John}]]]$
Quantifiers

- **Example:** \([S [NP [Q every][N man]][VP [v likes][NP John]]]\)
  
  - **Word Expression**
  - **every** \(\lambda P_1. [\lambda P_2. [\forall X (P_1(X) \rightarrow P_2(X))]\]
  - **man** man
  - **likes** \(\lambda Y. [\lambda X. [X \text{ likes } Y]]\)
  - **John** John

- **Steps:**
  \([Q \text{ every}][N \text{ man}] \rightarrow \lambda P_1. [\lambda P_2. [\forall X (P_1(X) \rightarrow P_2(X))]\](man)
  \([Q \text{ every}][N \text{ man}] \rightarrow \lambda P_2. [\forall X (\text{man}(X) \rightarrow P_2(X))]\]
  \([VP [v \text{ likes}][NP \text{ John}]] \rightarrow \lambda Y. [\lambda X. [X \text{ likes } Y]]\)(John)
  \([VP [v \text{ likes}][NP \text{ John}]] \rightarrow \lambda X. [X \text{ likes } John]\)
  \([S [NP [Q \text{ every}][N \text{ man}]]][VP [v \text{ likes}][NP \text{ John}]]\]

\(\lambda P_2. [\forall X (\text{man}(X) \rightarrow P_2(X))]\)(\(\lambda X. [X \text{ likes } John]\))
\(\forall X (\text{man}(X) \rightarrow \lambda X. [X \text{ likes } John](X))\)
Quantifiers

• **Example:** $\left[ S \left[ \begin{array}{l} \text{NP} \\ \text{Q every} \\ \text{N man} \end{array} \right] \left[ \begin{array}{l} \text{VP} \\ \text{V likes} \\ \text{NP John} \end{array} \right] \right]$
  
  – **Word**
  – **every** $\vdash (P_1, \vdash P_2)$.
  – **man** $\text{man}(X)$.
  – **likes** $\text{likes}(X,Y)$.
  – **John** $\text{john}$

• **Steps (Prolog-style):**
  
  $\left[ \begin{array}{l} \text{Q every} \\ \text{N man} \end{array} \right]$
  $\left[ \begin{array}{l} \text{NP} \\ \text{Q every} \\ \text{N man} \end{array} \right]$

  $\left[ \begin{array}{l} \text{V likes} \\ \text{NP John} \end{array} \right]$
  $\left[ \begin{array}{l} \text{VP} \\ \text{V likes} \\ \text{NP John} \end{array} \right]$

  $\left[ \begin{array}{l} \text{NP Q every} \\ \text{N man} \end{array} \right]$
  $\left[ \begin{array}{l} \text{VP V likes} \\ \text{NP John} \end{array} \right]$

  $\left[ \begin{array}{l} \text{S NP Q every} \\ \text{N man} \end{array} \right]$
  $\left[ \begin{array}{l} \text{VP V likes} \\ \text{NP John} \end{array} \right]$

  $S = \vdash (\text{man}(X), \vdash \text{likes}(X, \text{john}))$

  (pass up saturated NP as the value for S)
Quantifiers

- **Example:** \[ [_{S} [_{NP} [_{Q} every]_{N} man]]_{VP} [_{V} likes]_{NP} John] \]

  - **Word**
  - **Expression**
  - *every* findall(U,P1,L1), findall(V,P2,L2), subset(L1,L2).
  - *man* man(M).
  - *likes* likes(A,B).
  - *John* john

- **Steps:**
  \[ [_{Q} every]_{N} man] = (findall(U,P1,L1), findall(V,P2,L2), subset(L1,L2)), N = man(M),
  arg(1,Q,FA1), arg(2,FA1,N), saturate1(FA1,X), saturate1(N,X).
  \[ [_{NP} [_{Q} every]_{N} man]] = findall(X, man(X), L1), findall(V, P2, L2), subset(L1,L2)
  (pass up saturated Q as the value for the NP)

  \[ [_{V} likes]_{NP} John] = V = likes(A,B), NP = john, saturate2(V, NP).

  \[ [_{VP} [_{V} likes]_{NP} John]] = VP = likes(A,john)
  (pass up saturated V as the value for the VP)

  \[ [_{NP} [_{Q} every]_{N} man]][_{VP} [_{V} likes]_{NP} John]]

  NP = (findall(X, man(X), L1), findall(V, P2, L2), subset(L1,L2)), VP = likes(A,john),
  arg(2,NP,C2), arg(1,C2,FA2), arg(2,FA2,VP), saturate1(FA2,Y), saturate1(VP,Z).
Quantifiers

- **Example:** \([S_{NP} [_{Q} every]_{N} man] [_{VP} \_ \_ \_ \_ \_ likes]_{NP} John])\]
  - **Word**
  - **Expression**
  - every  \(\text{findall}(U,P1,L1), \text{findall}(V,P2,L2), \text{subset}(L1,L2))\.
  - man  \(\text{man}(M))\.
  - likes  \(\text{likes}(A,B))\.
  - John  \(\text{john})\.

- **Steps:**

\([_{NP} [_{Q} every]_{N} man] [_{VP} \_ \_ \_ \_ \_ likes]_{NP} John])\]
  
  \[?- NP = (\text{findall}(X, \text{man}(X), L1), \text{findall}(V, P2, L2), \text{subset}(L1, L2)), \text{VP} = \text{likes}(A, \text{john}), \text{arg}(2, NP, C2), \text{arg}(1, C2, FA2), \text{arg}(2, FA2, VP), \text{saturate1}(FA2, Y), \text{saturate1}(VP, Z).\]

\([S_{NP} [_{Q} every]_{N} man] [_{VP} \_ \_ \_ \_ \_ likes]_{NP} John])\]

\[S = \text{findall}(X, \text{man}(X), L1), \text{findall}(Y, \text{likes}(Y, \text{john}), L2), \text{subset}(L1, L2)\]

(pass up saturated **NP** as the value for **S**)
Names as Generalized Quantifiers

- In earlier lectures, we mentioned that names directly refer.
- Here is another idea.
- Conjunction
  - \( X \text{ and } Y \)
    - both \( X \) and \( Y \) have to be of the same type
    - in particular, semantically...
    - we want them to have the same semantic type
- what is the semantic type of every baby?

**Example**

- every baby and John likes ice cream
  - \([\text{NP} [\text{NP} \text{ every baby}] \text{ and } [\text{NP} \text{ John}] \text{ likes ice cream}]\)
  - every baby likes ice cream
  - \( \{X: \text{baby}(X)\} \subseteq \{Y: \text{likes}(Y, \text{ice cream})\} \)
  - John likes ice cream
  - \( ??? \subseteq \{Y: \text{likes}(Y, \text{ice cream})\} \)
  - John \( \in \{Y: \text{likes}(Y, \text{ice cream})\} \)
  - want everything to be a set (to be consistent)
  - i.e. want to state something like
  - \( \{X: \text{baby}(X)\} \cup \{X: \text{john}(X)\} \subseteq \{Y: \text{likes}(Y, \text{ice cream})\} \)
  - note: set union (\( \cup \)) is the translation of “and”
Negative Polarity Items

• **Negative Polarity Items (NPIs)**

• **Examples:**
  - every, any

• **Constrained distribution:**
  - have to be *licensed* in some way
  - grammatical in a “negated environment” or “question”

• **Examples:**
  - (13a) Shelby won’t *ever* bite you
  - (13b) Nobody has *any* money
  
  - (14a) *Shelby will *ever* bite you
  - (14b) *Noah has any money

  - * = ungrammatical

  - (15a) Does Shelby *ever* bite?
  - (15b) Does Noah have *any* money?
Negative Polarity Items

• Inside an *if-clause*:
  – (16a) *If* Shelby *ever* bites you, I’ll put him up for adoption
  – (16b) *If* Noah has *any* money, he can buy some candy

• Inside an *every-NP*:
  – (17a) *Every* dog which has *ever* bitten a cat feels the admiration of other dogs
  – (17b) *Every* child who has *any* money is likely to waste it on candy

• Not inside a *some-NP*:
  – (17a) *Some* dog which has *ever* bitten a cat feels the admiration of other dogs
  – (17b) *Some* child who has *any* money is likely to waste it on candy

Not to be confused with free choice (FC) *any* (meaning: ∀): *any man can do that*