LING 364: Introduction to Formal Semantics

Lecture 19

March 20th

Administrivia

- Handout: Chapter 6
  - Quantifiers
  - hard topic
  - *we'll start on it today*
- Read it for next Tuesday
  - Short Quiz 5

Administrivia

- We'll review Homework 4 next time
  - a bit behind on grading...

Leftover from Last Lecture

- Example:
  - (29) Only John loves his mother
  - (29') Only John **doesn't** love his mother
- World 1 for (29) (=31):
  - `loves(john,mother(john)).`
  - also, no other facts in the database that would satisfy the query
  - `?- loves(X,mother(john)), \+ X=john.`
- World 2 for (29) (=32):
  - `loves(john,mother(john)).`
  - also no other facts in the database that would satisfy the query
  - `? - loves(X,mother(X)),\+ X=john.`
- Both Worlds are possible since (29) is ambiguous
- Which one is preferred?

Leftover from Last Lecture

- **Example**:
  - (29) Only John loves his mother
  - (29') Only John **doesn't** love his mother
- **World 3 for (29'):**
  - `loves_not(john,mother(john)).`
  - also, no other facts in the database that would satisfy the query
  - `?- loves_not(X,mother(john)), \+ X=john.`
- **World 4 for (29'):**
  - `loves_not(john,mother(john)).`
  - also no other facts in the database that would satisfy the query
  - `? - loves_not(X,mother(X)),\+ X=john.`
- Both Worlds are possible since (*presumably*) (29') is also ambiguous like (29)
- Which one is preferred?

Today’s Topic

- Chapter 6: Quantifiers

Quantifiers

- Not all noun phrases (NPs) are (by nature) directly referential like names
- **Quantifiers**:
  - "*something to do with indicating the quantity of something*"
- **Examples**:
  - every child
  - nobody
  - two dogs
  - several animals
  - most people

  - nobody has seen a unicorn
  - could simply means *something like* (*Prolog-style*):
  - ?- findall(X,(person(X), seen(X,Y), unicorn(Y)),Set),length(Set,0).

Quantifiers

- Recall: compositionality idea:
  - *elements of a sentence combine in piecewise fashion to form an overall (propositional) meaning for the sentence*
- Example:
  - (4) Every baby cried

Word	Meaning
cried	cried(X).
baby	baby(X).
every	**?**
every baby cried	*proposition* (True/False)
	that can be evaluated in a given world

Quantifiers

- Scenario (Possible World):
  - suppose there are three babies...
    - baby(noah).
    - baby(merrill).
    - baby(dani).
  - all three cried
    - cried(noah).
    - cried(merrill).
    - cried(dani).
  - only Dani jumped
    - jumped(dani).
  - Noah and Dani swam
    - swam(noah).
    - swam(dani).

(6)	every baby	exactly one baby	most babies
cried	✓		✓
jumped		✓	
swam			✓

- **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

Quantifiers

- **think of quantifiers as "properties-of-properties"**
  - every_baby(P) **true** for P=cried
  - every_baby(P) **false** for P=jumped and P=swam
- **Generalized Quantifiers** (scary jargon alert!)
  - the idea that quantified NPs represent sets of sets
  - *this idea is not as wierd as it sounds*
  - we know
    - every_baby(P) is true for certain properties
  - view
    - every_baby(P) = set of all properties P for which this is true
  - in our scenario
    - every_baby(P) = {cried}
  - we know *cried* can also be view as a set itself
    - cried = set of individuals who cried
  - in our scenario
    - cried = {noah, merrill, dani}

Quantifiers

- **how do we define the expression every_baby(P)?**
- **(Montague-style)**
- every_baby(P) is shorthand for
  - for all individuals X, baby(X) -> P(X)
  - -> : *if-then (implication* : logic symbol*)*
- written another way (*lambda calculus-style*):
  - λP.[∀X.[baby(X) -> P(X)]]
  - ∀: *for all* (*universal quantifier*: logic symbol)

- **Example**:
  - every baby walks
    - for all individuals X, baby(X) -> walks(X)
  - more formally
  - $[_{NP}$ every baby] $[_{VP}$ walks]
    - λP.[∀X.[baby(X) -> P(X)]](walks)
    - ∀X.[baby(X) ->walks(X)]

Quantifiers

- **how do we define this Prolog-style?**
- **Example**:
  - every baby walks
  - $[_{NP}$ every baby$]$ $[_{VP}$ walks$]$
    - λP.[∀X (baby(X) -> P(X))](walks)
    - ∀X (baby(X) ->walks(X))
- **Possible World (Prolog database):**
  - :- dynamic baby/1. (*allows us to modify the baby database online*)
  - baby(a). baby(b).
  - walks(a). walks(b). walks(c).
  - individual(a). individual(b). individual(c).
- **What kind of query would you write?**
- **One Possible Query (*every* means there are *no exceptions*):**
  - ?- \+ (baby(X), \+ walks(X)). (**NOTE**: need a space between \+ and (here)
  - Yes (TRUE)
  - ?- baby(X), \+ walks(X).
  - No
  - ?- assert(baby(d)).
  - ?- baby(X), \+ walks(X).
  - X = d ;
  - Yes

using idea that ∀X P(X)
is the same as ¬∃X ¬P(X)
∃ = "there exists" (quantifier)
(**implicitly**: *all Prolog variables are existentially quantified variables*)

Aside: *Truth Tables*

- **logic of implication**
- P -> Q = (*truth value*)
- T T T
- F T T
- F F T
- T F F
- i.e. if P is true, Q must be true in order for P->Q to be true
- if P is false, doesn't matter what Q is, P->Q is true
- conventionally written as:

P	->	Q
T	T	T
F	T	T
F	T	F
T	F	F

P->Q=F only when P=T and Q=F

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

PvQ=F only when both P and Q are F

¬P	v	Q
TF	T	T
FT	T	T
FT	F	F
TF	T	T

¬PvQ=F only when P=T and Q=F

Hence, P->Q is equivalent to ¬PvQ

Aside: *Truth Tables*

- De Morgan's Rule
- $\neg(P \vee Q) = \neg P \wedge \neg Q$

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

¬(PvQ)
F
F
T
F

¬(PvQ)=T only when both P and Q are F

¬P	∧	¬Q
FT	F	FT
TF	F	FT
TF	T	TF
FT	F	FT

¬P∧¬Q=T only when both P and Q are F

Hence, ¬(PvQ) is equivalent to ¬P∧¬Q

Conversion into Prolog

Note: \+ (baby(X), \+walks(X)) is Prolog for ∀X (baby(X) -> walks(X))

Steps:

- ∀X (baby(X) -> walks(X))
- ∀X (¬baby(X) v walks(X))
  - (since P->Q = ¬PvQ, see truth tables from two slides ago)
- ¬∃X ¬ (¬baby(X) v walks(X))
  - (since ∀X P(X) = ¬∃X ¬P(X), no exception idea from 3 slides ago)
- ¬∃X (baby(X) ∧¬walks(X))
  - (by De Morgan's rule, see truth table from last slide)
- ¬(baby(X) ∧¬walks(X))
  - (can drop ∃X since all Prolog variables are basically existentially quantified variables)
- \+ (baby(X) ∧ \+walks(X))
  - (\+ = Prolog negation symbol)
- \+ (baby(X), \+walks(X))
  - (, = Prolog conjunction symbol)

Quantifiers

- **how do we define this Prolog-style?**
- **Example**:
  - every baby walks
  - $[_{NP}$ every baby$]$ $[_{VP}$ walks$]$
    - λP.[∀X.[baby(X) -> P(X)]](walks)
    - ∀X.[baby(X) ->walks(X)]
- **Another Possible World (Prolog database):**
  - :- dynamic baby/1.
  - :- dynamic walks/1.
  - % no facts (% = comment)
- **Does** ?- \+ (baby(X), \+ walks(X)). **still work?**

- **Yes because**
  - ?- baby(X), \+ walks(X).
  - No
- cannot be satisfied

Quantifiers

- **how do we define the expression every_baby(P)?**
- **(Montague-style)**
- every_baby(P) is shorthand for
  - $\lambda P.[\forall X.baby(X) \rightarrow P(X)]$

- **(Barwise & Cooper-style)**
- think directly in terms of sets
- *leads to another way of expressing the Prolog query*

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

- **Subset relation** (⊆)
- {X: baby(X)} ⊆ {X: walks(X)} *the "baby" set must be a subset of the "walks" set*

Quantifiers

- **(Barwise & Cooper-style)**
- think directly in terms of sets
- *leads to another way of expressing the Prolog query*

- **Example**: every baby walks
- {X: baby(X)} ⊆ {X: walks(X)} *the "baby" set must be a subset of the "walks" set*

- **How to express this as a Prolog query?**

  - **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*

Need a Prolog definition of the subset relation. This one, for example:

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

Quantifiers

- **Example**: every baby walks
- {X: baby(X)} ⊆{X: 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|_ ],L) :- member(X,L).
member(X,[X|_ ]).
member(X,[ _|L]) :- member(X,L).
```

- **Prolog Query:**
- ?- findall(X,baby(X),L1), findall(X,walks(X),L2), subset(L1,L2).

- **True for world:**
  - baby(a). baby(b).
  - walks(a). walks(b). walks(c).

L1 = [a,b]
L2 = [a,b,c]
?- subset(L1,L2) 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]
?- subset(L1,L2) is false