LING 364: Introduction to Formal Semantics

Lecture 6

January 31st

Administrivia

- this Thursday
  - (3:30pm – 4:45pm) lecture in Comm 214
  - (4:45pm – 5:45pm) **EXTRA** lab class in Social Sciences Lab 224

Administrivia

- Lab Class schedule for February
- Room:
  - SS 224

Calendar February 2006

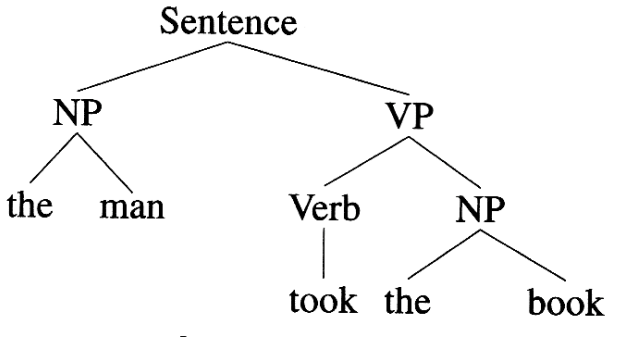
Tuesday	Wednesday	Thursday
	1 13:00 - 16:00 (ANTH537)	2 14:00 - 15:15 (LING439/53) ~~15:30 - 16:45 (HIST 396A)~~ 16:45 - 17:45 (LING364)
7 14:00 - 15:15 (LING439/53) 15:30 - 18:00 (ANTH696A)	8 13:00 - 16:00 (ANTH537)	9 14:00 - 15:15 (LING439/53) ~~15:30 - 16:45 (HIST 396A)~~ 16:45 - 17:45 (LING364)
14 14:00 - 15:15 (LING439/53) 15:30 - 18:00 (ANTH696A)	15 13:00 - 16:00 (ANTH537)	16 ~~14:00 - 15:15 (LING439/53)~~ 15:30 - 16:45 (LING364)
21 14:00 - 15:15 (LING439/53) 15:30 - 18:00 (ANTH696A)	22 13:00 - 16:00 (ANTH537)	23 ~~14:00 - 15:15 (LING439/53)~~ 15:30 - 16:45 (LING364)
28 14:00 - 15:15 (LING439/53) 15:30 - 18:00 (ANTH696A)		

AFTER CLASS

LAB CLASS

Last Time

- **Idea of a derivation:**
  - **top-down: expand**
    - from the start non-terminal to words
  - **bottom-up: reduce**
    - from words (sentence) to the start non-terminal
- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book

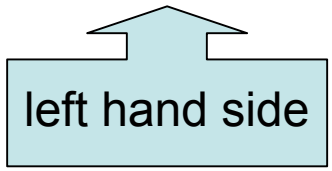


- **derivation**
  - **bottom-up (*one of many*)**

1. the man took the book
2. NP took the book
3. NP Verb the book
4. NP Verb NP
5. NP VP
6. Sentence

Last Time

- **formally**:
- *a grammar contains the following 4 things*
  - <N,T,P,S>
    - a set of non-terminal symbols (N)
    - a set of terminal symbols (T)
    - production rules (P) of the form
    - a designated start symbol (S)
- **from the example**
  - Non-terminals: {Sentence,VP,NP,Verb}
  - Terminals: {the,man,book,took}
  - Start symbol: Sentence
  - Production rules: *set of LHS → RHS rules*



- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book

Last Time

- Definite Clause Grammars (DCG)
  - *Prolog's built-in grammar rule facility*
- **format**
  - terminals and non-terminal symbols begin with lowercase letters
    - e.g. `sentence`, `vp`, `np`, `book`, `took`
    - *Note: variables begin with an uppercase letter (or underscore)*
  - `-->`
    - is the rewrite symbol
  - terminals are enclosed in square brackets to distinguish them from non-terminals (*list notation*)
    - e.g. `[the], [book], [took]`
  - comma (`,`) is the concatenation symbol
  - semicolon (`;`) is the disjunction symbol
  - a period (`.`) is required at the end of all DCG rules

Last Time

- **example**
  - Sentence → NP VP
  - VP → Verb NP
  - Verb → took
  - NP → the man
  - NP → the book

- **query:**

```
?- sentence(S,[]).
```

- `S` = sentence (as a list)
- `[]` = empty list
- i.e. *call the start symbol as a predicate and supply two arguments, a list and an empty list*

- **Prolog DCG version**

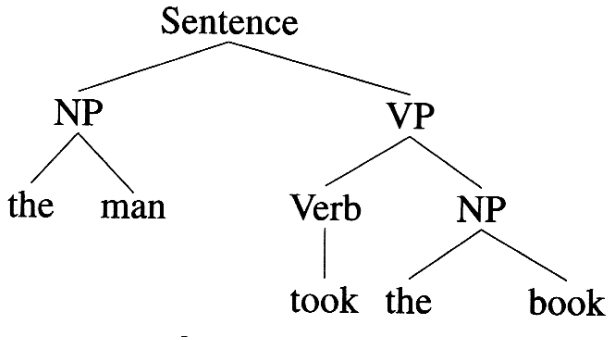
```
sentence --> np, vp.
vp --> verb, np.
verb --> [took].
np --> [the], [man].
np --> [the], [book].
```

- **Important Notes**
  - don't enter these rules into the database using assert/1.
  - **Use a file**.
- **Prolog List notation**:
  - [the,man,took,the,book]
  - *comma-separated terminals*

Did you try out the grammar on your computer?
Practice session: Computer Lab, Thursday

Prolog Grammar Rules

- **example**
  - `sentence --> np, vp.`
  - `vp --> verb, np.`
  - `verb --> [took].`
  - `np --> [the], [man].`
  - `np --> [the], [book].`



Last Time

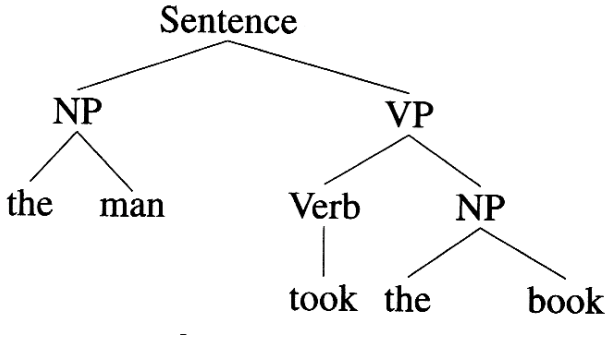
example queries

- `?- sentence([the,man,took,the,book],[]).`
- `Yes`
- means “the man took the book” ***is a member of the language generated by the grammar***

- `?- sentence([took,the,book],[]).`
- `No`
- means “took the book” ***is not in*** the grammar or “took the book” ***is not generated*** by the grammar

other start symbols

- `?- vp([took,the,book],[]).`
- `Yes`
- `?- np([the,book],[]).`
- `Yes`



Last Time

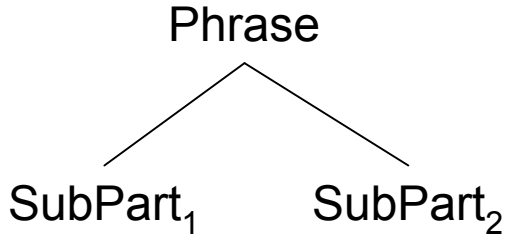
- **Chapter 2:**
  - *Putting a Meaning Together from Pieces*
- **Idea**:
  - each phrase or word contributes something to the sentence
  - notion of an **incomplete proposition**
- **example**

Word or Phrase	**Meaning**
Shelby barks	barks(shelby).
barks	barks(X). (**unsaturated proposition**)
Shelby	shelby

- **predication**:
  - *barks* is "***predicated of***" *shelby*, i.e. barks(X) and X = *shelby*

Last Time

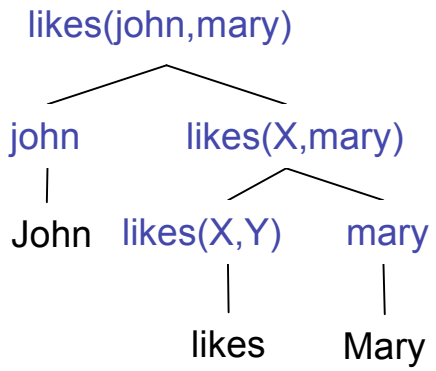
- Language exhibits (discrete) infinity and creativity (new phrases)
- **Principle of Compositionality**
  - Meaning(Phrase) =
    composition of
    Meaning($SubPart_1$),
    Meaning($SubPart_2$)
    and so on...



  - means meaning of sentence is derived systematically from the meaning of the words contained in that sentence

Quiz Answer

- given “John likes Mary” corresponds to likes(john,mary).
- meaning fragments are
  - **word or phrase** **meaning**
  - John john
  - likes Mary likes(X,mary).
  - likes likes(X,Y).
  - Mary mary
- here
  - each word contributes something to the meaning

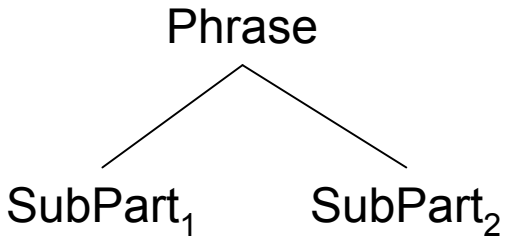


- How about?

  It is raining

Compositionality

- **Principle of Compositionality**
  - Meaning(Phrase) =
    composition of
    Meaning($SubPart_1$),
    Meaning($SubPart_2$)
    and so on...

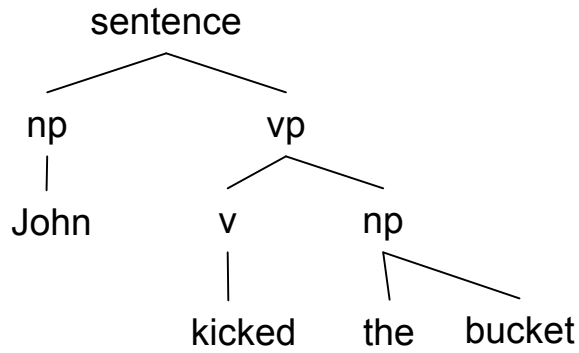


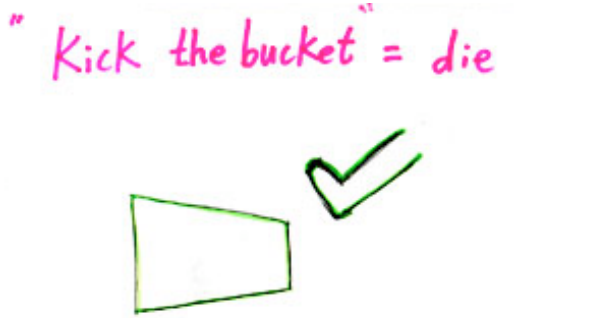
  - means meaning of sentence is derived systematically from the meaning of the words contained in that sentence

When does language violate compositionality?

Compositionality

- **idioms**
- example:
  - John kicked the bucket
  - **literal meaning**:
    - **word** **meaning**
    - john john
    - kick kick(X,Y).
    - bucket bucket
  - **idiomatic meaning**:
    - **word** **meaning**
    - john
    - kick
    - bucket





humanities.byu.edu/.../ kick_the_bucket.html

How about “John kicked a bucket”?

Compositionality

http://www.worldwidewords.org/qa/qa-kic1.htm

KICK THE BUCKET

[Q] *From Fred*: "Could you please tell me where the phrase *kick the bucket* originated?"

[A] There are two main theories about this one. One suggests that the word doesn't refer to our modern bucket at all, but to a sixteenth century word that comes from the French *buque*, meaning a yoke or similar piece of wood. It is said that the word was applied in particular to the beam from which a pig was hung in order to be slaughtered. Inevitably, the pig would struggle during the process, and would kick the *buque*.

The expression is attested to in particular by a citation in the *Oxford English Dictionary*: "The beam on which a pig is suspended after he has been slaughtered is called in Norfolk, even in the present day, a 'bucket'. Since he is suspended by his heels, the phrase to 'kick the bucket' came to signify to die" (I can't give you a date, as the editors just say it comes from a "modern newspaper", a rather sniffy annotation they used a century ago for sources not considered quite kosher. But it was probably in the 1890s).

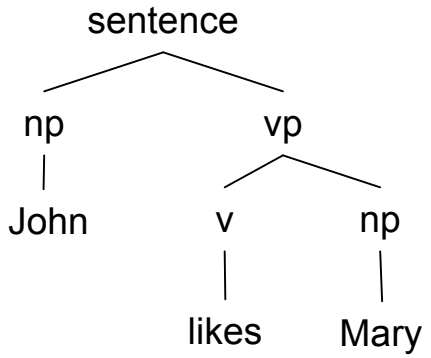
The other explanation, much less credible, is that the bucket is the one on which a suicide stands when hanging himself—kick away the bucket and the job is done. I've even seen the story attached specifically to the sad end of an ostler working at an inn on the Great North Road out of London. Don't believe a word of it.

Quiz Answers

- possible syntactic structures:
  1. $[_{S/S\text{-}bar}$ who $[_{VP}$ is $[_{NP}$ $[_{NP}$ a student] and $[_{NP}$ not $[_{NP}$ a baseball fan]]]]]?
  2. $[_{S/S\text{-}bar}$ who $[_{VP}$ is $[_{NP}$ not $[_{NP}$ $[_{NP}$ a baseball fan] or $[_{NP}$ a student]]]]]?
- **critical thing**:
  - notice the "scope" of *not*:
    1. *not* has scope over *a baseball fan*
    2. *not* has scope over *a baseball fan or a student*
- "scope" is an important notion in semantics
  - *and we indicate scope using syntactic structure*

Computing Phrase Structure

- **DCG:**
  - `sentence --> np, vp.`
  - `vp --> v, np.`
  - `v --> [likes].`
  - `np --> [john].`
  - `np --> [mary].`
- Query: (two argument version)
  - `?- sentence([john,likes,mary],[]).`
  - `Yes` (Answer)
- ***add in* phrase structure:**
  - `sentence(sentence(NP,VP)) --> np(NP), vp(VP).`
  - `vp(vp(V,NP)) --> v(V), np(NP).`
  - `v(v(likes)) --> [likes].`
  - `np(np(john)) --> [john].`
  - `np(np(mary)) --> [mary].`
- Query: (note: takes 3 arguments instead of 2)
  - `?- sentence(PS,[john,likes,mary],[]).`
  - `PS = sentence(np(john),vp(v(likes),np(mary)))`



Notation:

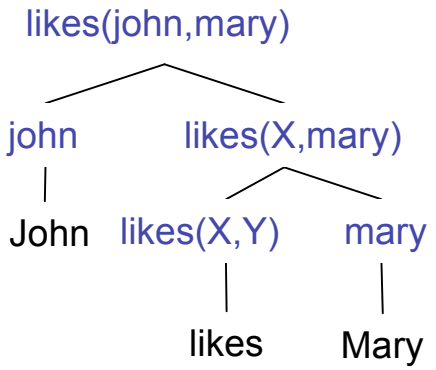
`np(mary)` = np | Mary

`v(likes)` = v | likes

and so on...

Computing Meaning

- **DCG:**
  - `sentence --> np, vp.`
  - `vp --> v, np.`
  - `v --> [likes].`
  - `np --> [john].`
  - `np --> [mary].`
- ***add in* phrase structure:**
  - `sentence(sentence(NP,VP)) --> np(NP), vp(VP).`
  - `vp(vp(V,NP)) --> v(V), np(NP).`
  - `v(v(likes)) --> [likes].`
  - `np(np(john)) --> [john].`
  - `np(np(mary)) --> [mary].`
- ***substitute* semantics *for* phrase structure:**
  - `sentence(P) --> np(NP1), vp(P), {saturate1(P,NP1)}.`
  - `vp(P) --> v(P), np(NP2), {saturate2(P,NP2)}.`
  - `v(likes(X,Y)) --> [likes].`
  - `np(john) --> [john].`
  - `np(mary) --> [mary].`



Notation:

`{ <Goal> }` means call Prolog `<Goal>`

add to Prolog Database:

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

arg/3 is a Prolog built-in:

arg(Nth,Predicate,Argument)
means make Nth argument of Predicate equal to Argument

examples:

```
?- arg(1,p(a,b,c),X).   X=a
?- arg(2,p(a,b,c),X).   X=b
?- arg(3,p(a,b,c),X).   X=c
?- arg(4,p(a,b,c),X).   No
```

Summary

- You now have the basic tools necessary to build:
  - phrase structure, and
  - meaning

  - (... *need some practice though*)
  - that's what the lab sessions are for...