Computational Semantics

CS224N 2004
Christopher Manning

(Borrows slides from Mary Dalrymple, Jason Eisner, and Jim Martin)

Why study computational semantics?

- Because everyone has been wanting me to talk about this all course!?
- Obvious applications
  - Summarization
  - Translation
  - Question answering
  - Information access
  - Talking to your pet robot
  - Speech user interfaces
- The next generation of intelligent applications need deeper semantics

Shallow vs. deep semantics

- We want to get from syntax to meaning!
- We can do more than we would have thought without deep linguistic analysis
- But we can't do everything we would like:
  - Not all tasks can ignore higher structure
  - Unsuitable if new text must be generated
  - Unsuitable if aim is not just to hand the user part of a document, relying on the author of the document and the user to make sense of the result

Example: Same words, different query

Kennedy assassination: Lee Harvey Oswald assassinated JFK, was shot to death several days later by Jack Ruby

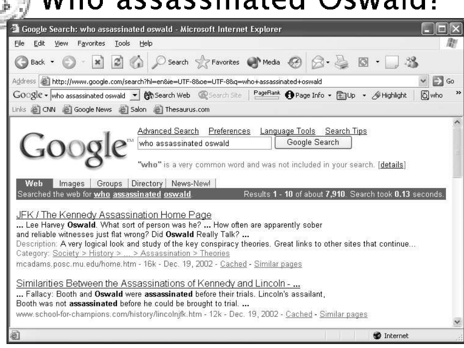
"Who did Oswald assassinate?"

answer: President Kennedy

"Who assassinated Oswald?"

answer: Jack Ruby

Who assassinated Oswald?



Who did Oswald assassinate?

Moral: You get what you pay for

- Cheap, fast, low-level techniques are useful, appropriate in domains where speed and volume are more important than accuracy
- More computationally expensive, higher-level techniques are appropriate when high-quality results are required

An early example: Chat-80

- Developed between 1979 and 1982 by Fernando Pereira and David Warren; became Pereira's dissertation
- Proof-of-concept natural language interface to database system
- Used in projects: e.g. Shoptalk (Cohen et al. 1989), a natural language and graphical interface for decision support in manufacturing
- Even used in an ANLP-2000 conference paper!
- Available in src directory
  - Need sicstus prolog: /afs/ir/class/symbsys139p/bin/sicstus

The CHAT-80 Database

```
% Facts about countries.
% country(Country,Region,Latitude,Longitude,
%     Area (sqmiles), Population, Capital,Currency)
country(andorra,southern_europe,42,-1,179,
   25000,andorra_la_villa,franc_peseta).
country(angola,southern_africa,-12,-18,481351,
   5810000,luanda,?).
country(argentina,south_america,-35,66,
   1072067, 23920000,buenos_aires,peso).

capital(C,Cap) :- country(C,_,_,_,_,_,Cap,_).
```

Chat-80 trace (illegibly small)

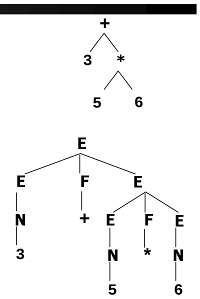
```
Question: What is the capital of
   Australia?
Parse: 0.0sec.
whq
  $VAR
    1
  s
    np
      3+sin
      wh(B)
      []
    verb(be,active,pres+fin,[],pos)
    arg
      dir
      np
        3+sin
        np_head
          det(the(sin))
          []
          capital
        pp
          prep(of)
          np
            3+sin
            name(australia)
            []
    []

Semantics: 0.0sec.
answer([B]) :-
   capital(australia,B)

canberra.
```

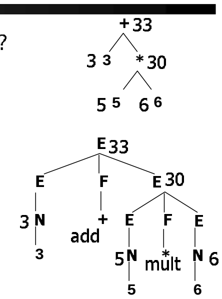
Getting semantics: programming language interpreter

- What is meaning of 3+5*6?
- First parse it into 3+(5*6)



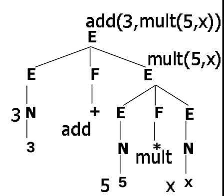
Programming Language Interpreter

- What is meaning of 3+5*6?
- First parse it into 3+(5*6)
- Now give a meaning to each node in the tree (bottom-up)



More complex meanings

- How about 3+5*x?
- Don't know x at compile time
- "Meaning" at a node is a piece of code, not a number



What Counts as Understanding?

- A somewhat difficult philosophical question
- We understand if we can respond appropriately
  - "throw axe at dwarf"
- We understand statement if we can determine its truth
- We understand statement if we can use it to answer questions [similar to above – requires reasoning]
  - **Easy:** John ate pizza. What was eaten by John?
- Understanding is the ability to translate
  - English to Chinese? requires deep understanding?
  - English to **logic**? deepest – the definition we'll use!
    - all humans are mortal = $\forall x$ [human(x) $\Rightarrow$mortal(x)]
- We assume we have logic-manipulating rules to tell us how to act, draw conclusions, answer questions ...

Lecture Plan

- Today:
  - Look at some sentences and phrases
  - What would be reasonable logical representations for them?
  - Get some idea of compositional semantics
- Wednesday:
  - How can we build those representations?
- Another course (somewhere in AI, hopefully):
  - How can we reason with those representations?

Logic: Some Preliminaries

Three major kinds of objects

1. Booleans (Bool)
   - Roughly, the semantic values of sentences
2. Individuals/Entities (Ind)
   - Values of NPs, i.e., objects
   - Maybe also other types of entities, like times
3. Functions of various types
   - A function returning a boolean is called a "predicate" – e.g., frog(x), green(x)
   - Functions might return other functions!
   - Function might take other functions as arguments!

Logic: Lambda Terms

- Lambda terms:
  - A way of writing "anonymous functions"
    - No function header or function name
    - But defines the key thing: **behavior** of the function
    - Just as we can talk about 3 without naming it "x"
  - Let square = λp p*p
  - Equivalent to int square(p) { return p*p; }
  - But we can talk about λp p*p without naming it
  - Format of a lambda term: λ variable . expression

Logic: Lambda Terms

- Lambda terms:
  - Let square = λp p*p
  - Then square(3) = (λp p*p)(3) = 3*3
  - Note: square(x) isn't a function! It's just the value x*x.
  - But **λx** square(x) = λx x*x = λp p*p = square
    (proving that these functions are equal – and indeed they are, as they act the same on all arguments: what is (λx square(x))(y)?)
  - Let even = λp (p mod 2 == 0) a predicate: returns true/false
  - even(x) is true if x is even
  - How about even(square(x))?
  - λx even(square(x)) is true of numbers with even squares
    - Just apply rules to get λx (even(x*x)) = λx (x*x mod 2 == 0)
    - This happens to denote the same predicate as even does

Logic: Multiple Arguments

- All lambda terms have one argument
- But we can fake multiple arguments ...
- Suppose we want to write times(5,6)
- Remember: square can be written as λx.square(x)
- Similarly, times is equivalent to λx.λy.times(x,y)
- Claim that times(5)(6) means same as times(5,6)
  - times(5) = (λx.λy.times(x,y)) (5) = λy.times(5,y)
    - If this function weren't anonymous, what would we call it?
  - times(5)(6) = (λy times(5,y))(6) = times(5,6)
- Referred to as "currying"

Logic: Interesting Constants

- We have "constants" that name some of the entities and functions (e.g., times):
  - GeorgeWBush – an entity
  - red – a predicate on entities
    - holds of just the red entities: red(x) is true if x is red!
  - loves – a predicate on 2 entities
    - loves(GeorgeWBush, LauraBush)
    - *Question:* What does loves(LauraBush) denote?
- Constants used to define meanings of words
- Meanings of phrases will be built from the constants

Logic: Interesting Constants

- most – a predicate on 2 predicates on entities
  - most(pig, big) = "most pigs are big"
    - Equivalently, most(λx pig(x), λx big(x))
  - returns true if most of the things satisfying the first predicate also satisfy the second predicate
- similarly for other quantifiers
  - all(pig,big) (equivalent to ∀x pig(x) ⇒ big(x))
  - exists(pig,big) (equivalent to ∃x pig(x) AND big(x))
  - can even build complex quantifiers from English phrases:
    - "between 12 and 75"; "a majority of"; "all but the smallest 2"

Quantifier Order

- Groucho Marx celebrates quantifier order ambiguity:
  - **In this country <u>a woman</u> gives birth <u>every 15 min</u>. Our job is to find that woman and stop her.**
  - ∃woman (∀15min gives-birth-during(woman, 15min))
  - ∀15min (∃woman gives-birth-during(15min, woman))
  - Surprisingly, both are possible in natural language!
  - Which is the joke meaning? (where it's always the same woman)

Compositional Semantics

- We've discussed what semantic representations should look like.
- **But how do we get them from sentences???**
- First – parse to get a syntax tree.
- Second – look up the semantics for each word.
- Third – build the semantics for each constituent
  - Work from the bottom up
  - The syntax tree is a "recipe" for how to do it
- Principle of Compositionality
  - The meaning of a whole is derived from the meanings of the parts, via composition rules

A simple grammar of English

(in Definite Clause Grammar, DCG, form – as in Prolog)

sentence --> noun_phrase, verb_phrase.
noun_phrase --> proper_noun.
noun_phrase --> determiner, noun.
verb_phrase --> verb, noun_phrase.

Proper_noun --> [John]	verb --> [ate]
Proper_noun --> [Mary]	verb --> [kissed]
determiner --> [the]	noun --> [cake]
determiner--> [a]	noun --> [lion]

Extending the grammar to check number agreement between subjects and verbs

S --> NP(Num), VP(Num).
NP(Num) --> Proper_noun(Num).
NP(Num) --> det(Num), noun(Num).
VP(Num) --> verb(Num), noun_phrase(_).

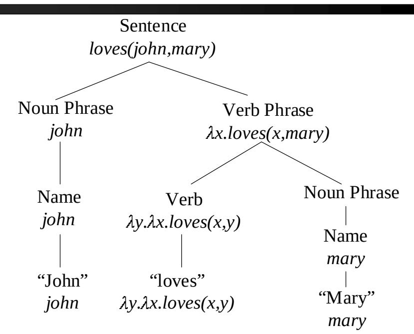
Proper_noun(s) --> [Mary].	noun(s) --> [lion].
det(s) --> [the].	noun(p) --> [lions].
det(p) --> [the].	verb(s) --> [eats].
	verb(p) --> [eat].

A simple DCG grammar with semantics

sentence(SMeaning) --> noun_phrase(NPMeaning), verb_phrase(VPMeaning), {combine (NPMeaning, VPMeaning, SMeaning)}.
verb_phrase(VPMeaning) --> verb(Vmeaning), noun_phrase(NPMeaning), {combine (NPMeaning, VMeaning, VPMeaning)}.
noun_phrase (NPMeaning) --> name(NPMeaning).

name(john) --> [john].	verb(λx.jumps(x)) --> [jumps]
name(mary) --> [mary].	verb(λy.λx.loves(x,y)) -->[loves]

Parse tree with associated semantics



Augmented CFG Rules

- We can also accomplish this just by attaching semantic formation rules to our syntactic CFG rules

$$A \rightarrow \alpha_1 ... \alpha_n \quad \{ f(\alpha_1.sem, ..., \alpha_n.sem) \}$$

- This should be read as the semantics we attach to A can be computed from some function applied to the semantics of A's parts.
- The functions/operations permitted in the semantic rules are restricted, falling into two classes
  - Pass the semantics of a daughter up unchanged to the mother
  - Apply (as a function) the semantics of one of the daughters of a node to the semantics of the other daughters