# Computational Semantics 2

CS224N
Christopher Manning

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

## Today

1. Quantifiers [15 mins]
2. Semantic Grammars [15 mins]
3. Introducing a formal grammar fragment [15 mins]
4. Model theory [5 mins]
5. Extending applying the formal fragment [30 mins]
6. Semantic Parsing [0 or more minutes]

## More NLP!

NLP in the Spring!

- CS224U: Natural Language Understanding
  - Much more detail of semantic and NLU. Great class.
  - Chris Potts, Bill MacCartney
- CS224D: Deep Learning for NLP
  - Neural nets and deep learning in much more detail
  - Richard Socher
- CS276: Information Retrieval and Web Search
  - Learn how Google works ☺. From infrastructure to ML.

## QUANTIFIERS

## How do things get more complex? (The former) GRE analytic section

- Six sculptures – C, D, E, F, G, H – are to be exhibited in rooms 1, 2, and 3 of an art gallery.
  - Sculptures C and E may not be exhibited in the same room.
  - Sculptures D and G must be exhibited in the same room.
  - If sculptures E and F are exhibited in the same room, no other sculpture may be exhibited in that room.
  - At least one sculpture must be exhibited in each room, and no more than three sculptures may be exhibited in any room.
- If sculpture D is exhibited in room 3 and sculptures E and F are exhibited in room 1, which of the following may be true?

1. Sculpture C is exhibited in room 1.
2. Sculpture H is exhibited in room 1.
3. Sculpture G is exhibited in room 2.
4. Sculptures C and H are exhibited in the same room.
5. Sculptures G and F are exhibited in the same room.

## How do things get more complex? (The former) GRE analytic section

- **Six** sculptures – C, D, E, F, G, H – are to be exhibited in rooms 1, 2, and 3 of an art gallery.
  - Sculptures C and E may not be exhibited in the same room.
  - Sculptures D and G must be exhibited in the same room.
  - If sculptures E and F are exhibited in the same room, **no other** sculpture may be exhibited in that room.
  - **At least one** sculpture must be exhibited in **each** room, and **no more than three** sculptures may be exhibited in **any** room.
- If sculpture D is exhibited in room 3 and sculptures E and F are exhibited in room 1, which of the following may be true?

1. Sculpture C is exhibited in room 1.
2. Sculpture H is exhibited in room 1.
3. Sculpture G is exhibited in room 2.
4. Sculptures C and H are exhibited in the same room.
5. Sculptures G and F are exhibited in the same room.

## Adding more complex NPs

NP: A man ~> ∃x.man(x)
S: A man loves Mary
~> * love(∃x.man(x),mary)

How do we fix this?

## A disappointment

Our first idea for NPs with determiner didn't work out:
"A man" ~> ∃z.man(z)
"A man loves Mary" ~> * love(∃z.man(z),mary)

But what was the idea after all?
Nothing!
∃z.man(z) just isn't the meaning of "a man".

If anything, it translates the complete sentence
"There is a man"

Let's try again, systematically…

## A solution for quantifiers

What we want is:
"A man loves Mary" ~> ∃z(man(z) ∧ love(z,mary))

What we have is:
"man" ~> λy.man(y)
"loves Mary" ~> λx.love(x,mary)

How about: ∃z(λy.man(y)(z) ∧ λx.love(x,mary)(z))
Remember: We can use variables for any kind of term.
So next:
λP(λQ.∃z(P(z) ∧Q(z))) <~ "A"

## Logic: Interesting Constants

- Quantifiers
- all – a predicate on two properties
  - all(dog)(bark) = "all dogs bark"
    - Equivalently, all(λx dog(x))(λx bark(x))
  - returns true if every individual that makes the first predicate true also satisfy the second predicate
- similarly for other quantifiers
  - all is a higher order function. It maps:
    - (Ind → Bool) → [(Ind → Bool) → Bool]
- Note that it's just part of our set of functions
  - Not some special funky thing like in predicate calculus

## Logic: Interesting Constants

- Generalized Quantifiers
- 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
  - at-least-2(pig,big) – equivalent to |pig(x) AND big(x)| > 2
  - Not-exists(pig,beautiful)
    - equivalent to ! ∃x pig(x) AND beautiful(x)
  - can even build complex quantifiers from English phrases:
    - "between 12 and 75"; "a minority 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)

## Scope Needs to be Resolved!

*At least one sculpture must be exhibited in each room.*
The same sculpture in each room?
*No more than three sculptures may be exhibited in any room.*

Reading 1: For every room, there are no more than three sculptures exhibited in it.

Reading 2: Only three or less sculptures are exhibited ( the rest are not shown).

Reading 3: Only a certain set of three or less sculptures may be exhibited in any room ( for the other sculptures there are restrictions in allowable rooms).

- Some readings will be ruled out by being uninformative or by contradicting other statements
- Otherwise we must be content with distributions over scope-resolved semantic forms

## SEMANTIC GRAMMARS

## An alternative: Semantic Grammars

- A problem with traditional linguistic grammars is that they don't necessarily reflect the semantics in a straightforward way
- You can deal with this by…
  - Fighting with the grammar
    - Complex lambdas and complex terms, etc.
  - Rewriting the grammar to reflect the semantics
    - And in the process give up on some syntactic niceties
    - known as "Semantic grammars"
      - Simple idea, dumb name

## Lifer Semantic Grammars

- Example domain—access to DB of US Navy ships
  S → <present> the <attribute> of <ship>
  <present> → what is | [can you] tell me
  <attribute> → length | beam | class
  <ship> → the <shipname>
  <shipname> → kennedy | enterprise
  <ship> → <classname> class ships
  <classname> → kitty hawk | lafayette
- Example inputs recognized by above grammar:
  *can you tell me the class of the Enterprise*
  *what is the length of Kitty Hawk class ships*
  - Many categories are not "true" syntactic categories
  - Words are recognized by their context rather than category (e.g. *class*)
  - Recognition is strongly directed
  - Strong direction useful for error detection and correction
    - G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J.Slocum. 1978. Developing a natural language interface to complex data. ACM Transactions on Database Systems 3:105-147

## Semantic Grammar

- The term semantic grammar refers to the motivation for the grammar rules
  - The technology (plain CFG rules with a set of terminals) is the same as we've been using
  - The good thing about them is that you get exactly the semantic rules you need
  - The bad thing is that you need to develop a new grammar for each new domain
- Typically used in conversational agents in constrained domains
  - Limited vocabulary
  - Limited grammatical complexity
  - Syntactic parsing can often produce all that's needed for semantic interpretation even in the face of "ungrammatical" input – write fragment rules

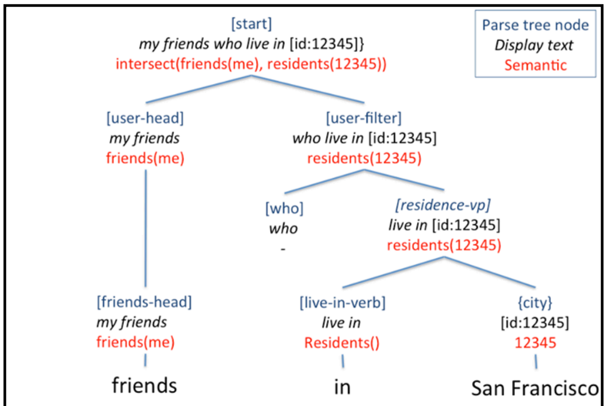
## Semantic Grammars Summary

- Advantages:
  - Efficient recognition of limited domain input
  - Absence of overall grammar allows pattern-matching possibilities for idioms, etc.
  - No separate interpretation phase
  - Strength of top-down constraints allows powerful ellipsis mechanisms
    *What is the length of the Kennedy? The Kittyhawk?*
- Disadvantages:
  - Different grammar required for each new domain
  - Lack of overall syntax can lead to "spotty" grammar coverage
    - E.g. fronting possessive in "<attribute> of <ship>" to <ship> 's <attribute> doesn't imply fronting in "<rank> of <officer>"
  - Difficult to develop grammars past a certain size
  - Suffers from fragility

## Facebook Graph Search

- Uses a weighted context free grammar (WCFG) to represent the Graph Search query language:
- [start] => [users]  $1
- [users] => my friend(s)  friends(me)
- [users] => friends of [users]  friends($1)
- [users] => {user}  $1
- [users] => {user} {user-filter}  intersect($1, $2)
- [start] => [photos]  $1
- [photos] => photos of [users]  photos($1)
- A terminal symbol can be an entity, e.g., {user}, {city}, {employer}, {group}; it can also be a word/phrase, e.g., *friends*, *live in*, *work at*, *members*, etc. A parse tree starts from [start] and expanding via rules to terminal symbols.

https://www.facebook.com/notes/facebook-engineering/under-the-hood-the-natural-language-interface-of-graph-search/101514327330489
http://spectrum.ieee.org/telecom/internet/the-making-of-facebooks-graph-sear




The parse tree, semantic and entity ID used in the above example are for illustration only; they do not represent real information used in Graph Search Beta