Computational Semantics
Chapter 18

Lecture #12

November 2012

We will not do all of this…

Semantic Analysis

- Semantic analysis is the process of taking in some linguistic input and producing a meaning representation for it.
  - There are many ways of doing this, ranging from completely ad hoc domain specific methods to more theoretically founded by not quite useful methods.
  - Different methods make more or less (or no) use of syntax
  - We're going to start with the idea that syntax does matter
    - The compositional rule-to-rule approach

Compositional Analysis

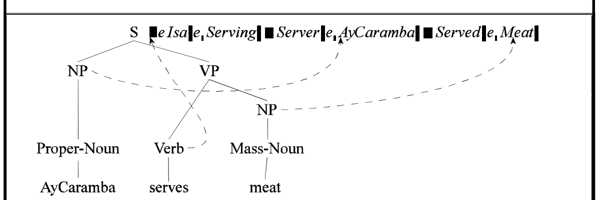
- Principle of Compositionality
  - The meaning of a whole is derived from the meanings of the parts
- What parts?
  - The constituents of the syntactic parse of the input.

Example

- AyCaramba serves meat.

$$\exists e\ Serving(e) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$$

Compositional Analysis



Compositional Semantics

- Note in the previous example:
- Part of the meaning derives from the people and activities it's about (predicates and arguments, or, nouns and verbs) and part from the way they are ordered and related grammatically: syntax
- Question: can we link up syntactic structures to a corresponding semantic representation to produce the 'meaning' of a sentence in the course of parsing it?

Specific vs. General-Purpose Rules

- We don't want to have to specify for every possible parse tree what semantic representation it maps to
- We want to identify general mappings from parse trees to semantic representations:
  - Again (as with feature structures) we will augment the lexicon and the grammar
  - Rule-to-rule hypothesis: a mapping exists between rules of the grammar and rules of semantic representation

Semantic Attachments

- Extend each grammar rule with instructions on how to map the components of the rule to a semantic representation (grammars are getting complex)

  S → NP VP {VP.sem(NP.sem)}

- Each semantic function is defined in terms of the semantic representation of choice
- Problem: how to define these functions and how to specify their composition so we always get the meaning representation we want from our grammar?

Augmented Rules

- Let's look at this a little more abstractly. Consider the general case of a grammar rule:

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

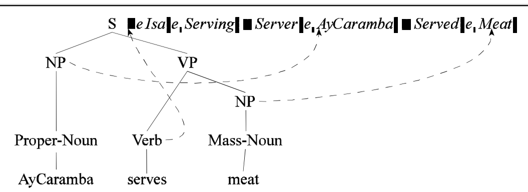
- This should be read as the semantics we attach to A can be computed from some function applied to the semantics of A's parts.

Augmented Rules

- As we'll see the class of actions performed by *f* in the following rule can be quite restricted.

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

Compositional Analysis



A 'Simple' Example

AyCaramba serves meat.

- Associating constants with constituents
  - ProperNoun → AyCaramba {AyCaramba}
  - MassNoun → meat {Meat}
- Defining functions to produce these from input
  - NP → ProperNoun {ProperNoun.sem}
  - NP → MassNoun {MassNoun.sem}
- Assumption: meaning reps of children are passed up to parents for non-branching constituents

- Verbs here are where the action is
  - V → serves { ∃ (e,x,y) Isa(e,Serving) ^ Server(e,x) ^ Served(e,y)}
  - Will every verb have its own distinct representation?
  - Predicate(Agent,Patient)…
- How do we combine these pieces?
  - VP → V NP {????}
  - Goal: ∃ (e,x) Isa(e,Serving) ^ Server(e,x) ^ Served(e,Meat)
  - S → NP VP {????}
  - Goal: ∃ (e) Isa(e,Serving) ^ Server(e, AyCaramba) ^ Served(e,Meat)
  - VP and S semantics must tell us
    - Which variables are to be replaced by which arguments?
    - How is this replacement done?

Lambda Notation

- Extension to FOPC
  - λ x P(x)
  - λ + variable(s) + FOPC expression in those variables
- Lambda binding
  - Apply lambda-expression to logical terms to bind lambda-expression's parameters to terms (lambda reduction)
  - Simple process: substitute terms for variables in lambda expression

    λxP(x)(car)

    P(car)

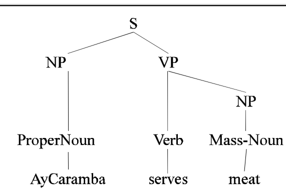
- Lambda notation provides requisite verb semantics
  - Formal parameter list makes variables within the body of the logical expression available for binding to external arguments provided by e.g. NPs
  - Lambda reduction implements the replacement
- Semantic attachment for grammar rules:
  - S → NP VP {VP.sem(NP.sem)}
  - VP → V NP {V.sem(NP.sem)}
  - V → serves {???}

  { ∃ (e,x,y) Isa(e,Serving) ^ Server(e,y) ^ Served(e,x)} becomes

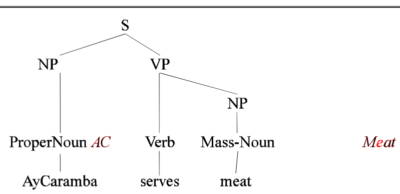
  {λy λx ∃ (e) Isa(e,Serving) ^ Server(e,x) ^ Served(e,y)}

  - Now 'x' is available to be bound when V.sem is applied to NP.sem, and 'y' is available to be bound when the S rule is applied.

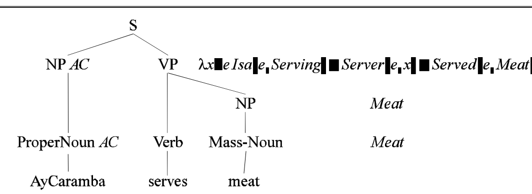
Example



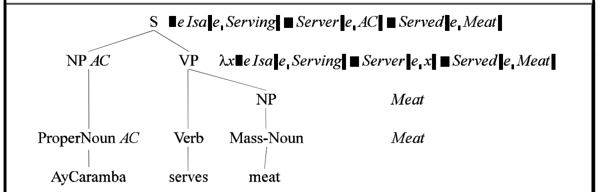
Example



Example



Example



Key Points

- Each node in a tree corresponds to a rule in the grammar
- Each grammar rule has a semantic rule associated with it that specifies how the semantics of the LHS of that rule can be computed from the semantics of its daughters.

Strong Compositionality

- The semantics of the whole is derived solely from the semantics of the parts.

  (i.e. we ignore what's going on in other parts of the tree).

Predicate-Argument Semantics

- The functions/operations permitted in the semantic rules fall 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

Mismatches

- There are unfortunately some annoying mismatches between the syntax of FOPC and the syntax provided by our grammars…
- So we'll accept that we can't always directly create valid logical forms in a strictly compositional way
  - We'll get as close as we can and patch things up after the fact.

Quantified Phrases

- Consider

  A restaurant serves meat.
- Assume that A restaurant looks like

$$\exists x\ Isa(x, Restaurant)$$

- If we do the normal lambda thing we get

$$\exists e Serving(e) \wedge Server(e, \exists x Isa(x, Restaurant)) \wedge Served(e, Meat)$$

Complex Terms

- Allow the compositional system to pass around representations like the following as objects with parts:

```
Complex-Term → <Quantifier var body>
```

$$< \exists\, x\ Isa(x, \text{Restaurant}) >$$

25

Example

- Our restaurant example winds up looking like

$$\exists e Serving(e) \wedge Server(e, < \exists x Isa(x, Restaurant) >) \wedge Served(e, Meat)$$

- Big improvement…

26

Conversion

- So… complex terms wind up being embedded inside predicates. So pull them out and redistribute the parts in the right way…

P(<quantifier, var, body>)
`turns into`
Quantifier var body connective P(var)

27

Example

$$Server(e, < \exists\, x\ Isa(x, \text{Restaurant}) >)$$
$$\Rightarrow$$
$$\exists\, x\ Isa(x, \text{Restaurant}) \wedge Server(e, x)$$

28

Quantifiers and Connectives

- If the quantifier is an existential, then the connective is an ^ (and)
- If the quantifier is a universal, then the connective is an -> (implies)

29

Multiple Complex Terms

- Note that the conversion technique pulls the quantifiers out to the front of the logical form…
- That leads to ambiguity if there's more than one complex term in a sentence.

30

Quantifier Ambiguity

- Consider
  - Every restaurant has a menu
  - Every restaurant has a beer.
  - I took a picture of everyone in the room.

  - That could mean that
    every restaurant has a menu
  - Or that
    There's some super-menu out there and all restaurants have that menu

31

Quantifier Scope Ambiguity

$$\forall x Restaurant(x) \Rightarrow$$
$$\exists e, y Having(e) \wedge Haver(e,x) \wedge Had(e,y) \wedge Isa(y, Menu)$$

$$\exists y Isa(y, Menu) \wedge \forall x Isa(x, Restaurant) \Rightarrow$$
$$\exists e Having(e) \wedge Haver(e,x) \wedge Had(e,y)$$

32

Ambiguity

- This turns out to be a lot like the prepositional phrase attachment problem
- The number of possible interpretations goes up exponentially with the number of complex terms in the sentence
- The best we can do is to come up with weak methods to prefer one interpretation over another

33

Doing Compositional Semantics

- To incorporate semantics into grammar we must
  - Figure out right representation for a single constituent based on the parts of that constituent (e.g. Adj)
  - Figure out the right representation for a category of constituents based on other grammar rules making use of that constituent (e.g Nom→ Adj Nom)
- This gives us a set of function-like semantic attachments incorporated into our CFG
  - E.g. Nom → Adj Nom {λx Nom.sem(x) ^ Isa(x,Adj.sem)}

34

What do we do with them?

- As we did with feature structures:
  - Alter an Early-style parser so when constituents (dot at the end of the rule) are completed, the attached semantic function is applied and a meaning representation created and stored with the state
- Or, let parser run to completion and then walk through resulting tree running semantic attachments from bottom-up

35

Option 1 (Integrated Semantic Analysis)

S → NP VP {VP.sem(NP.sem)}
  - VP.sem has been stored in state representing VP
  - NP.sem has been stored with the state for NP
  - When rule completed, go get value of VP.sem, go get NP.sem, and apply VP.sem to NP.sem
  - Store result in S.sem.
- As fragments of input parsed, semantic fragments created
- Can be used to block ambiguous representations

36

Drawback

- You also perform semantic analysis on orphaned constituents that play no role in final parse
- Hence, case for pipelined approach: Do semantics ***after*** syntactic parse
- But….

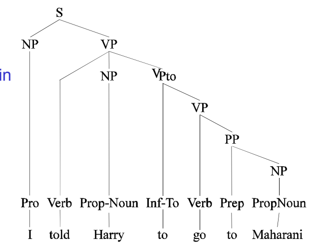
- Let's look at some other examples….

37

Harder Example

- What makes this hard?

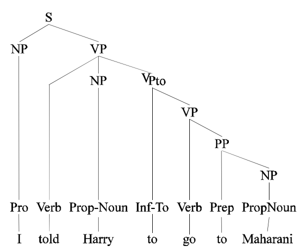
- What role does Harry play in all this?



38

Harder Example

∃ e,f,x Isa(e, Telling) ^
Isa(f, Going) ^
Teller(e, Speaker) ^
Tellee(e, Harry) ^
ToldThing(e, f) ^
Goer(f, Harry) ^
Destination(f, x)



39

Harder Example

- The VP for told is VP -> V NP VPto
  - So you do what?
    - Apply the semantic function attached to VPTO the semantics of the NP; this binds Harry as the goer of the going.
    - Then apply the semantics of the V to the semantics of the NP; this binds Harry as the Tellee of the Telling
    - And to the result of the first application to get the right value of the told thing.

    V.Sem(NP.Sem, VPto.Sem(NP.Sem))

40

Harder Example

- That's a little messy and violates the notion that the grammar ought not to know much about what is going on in the semantics…
- Better might be
  - V.sem(NP.Sem, VPto.Sem) ????
  - VPto.sem(V.sem, NP.sem)???
  - i.e Apply the semantics of the head verb to the semantics of its arguments.
  - Complicate the semantics of the verb inside VPto to figure out what's going on.

41

Two Philosophies

1. Let the syntax do what syntax does well and don't expect it to know much about meaning
   - In this approach, the lexical entry's semantic attachments do the work
2. Assume the syntax does know about meaning
   - Here the grammar gets complicated and the lexicon simpler

42

Example

- Consider the attachments for the VPs

  VP -> Verb NP NP (gave Mary a book)

  VP -> Verb NP PP (gave a book to Mary)

  Assume the meaning representations should be the same for both. Under the lexicon-heavy scheme the attachments are:

  VP.Sem(NP.Sem, NP.Sem)

  VP.Sem(NP.Sem, PP.Sem)

43

Example

- Under the syntax-heavy scheme we might want to do something like
- VP -> V NP NP

  V.sem ^ Recip(NP1.sem) ^ Object(NP2.sem)
- VP -> V NP PP

  V.Sem ^ Recip(PP.Sem) ^ Object(NP1.sem)
- I.e. the verb only contributes the predicate, the grammar "knows" the roles.

44

Integration

- Two basic approaches
  - Integrate semantic analysis into the parser (assign meaning representations as constituents are completed)
  - Pipeline… assign meaning representations to complete trees only after they're completed

45

Example

- From BERP
  - I want to eat someplace near campus
  - Somebody tell me the two meanings…

46

Pros and Cons

- If you integrate semantic analysis into the parser as its running…
  - You can use semantic constraints to cut off parses that make no sense
  - You assign meaning representations to constituents that don't take part in the correct (most probable) parse

47

Non-Compositionality

- Unfortunately, there are lots of examples where the meaning (loosely defined) can't be derived from the meanings of the parts
  - Idioms, jokes, irony, sarcasm, metaphor, metonymy, indirect requests, etc

48

English Idioms

- Kick the bucket, buy the farm, bite the bullet, run the show, bury the hatchet, etc…
- Lots of these… constructions where the meaning of the whole is either
  - Totally unrelated to the meanings of the parts (kick the bucket)
  - Related in some opaque way (run the show)

Example

- Enron is the tip of the iceberg.
  NP -> "the tip of the iceberg"
- Not so good… attested examples…
  - the tip of Mrs. Ford's iceberg
  - the tip of a 1000-page iceberg
  - the merest tip of the iceberg
- How about
  - That's just the iceberg's tip.

Example

- What we seem to need is something like
- NP ->
  An initial NP with tip as its head followed by
  a subsequent PP with of as its head and that has iceberg as the head of its NP
  And that allows modifiers like merest, Mrs. Ford, and 1000-page to modify the relevant semantic forms

The Tip of the Iceberg

- Describing this particular construction
  1. A fixed phrase with a particular meaning
  2. A syntactically and lexically flexible phrase with a particular meaning
  3. A syntactically and lexically flexible phrase with a partially compositional meaning

Constructional Approach

- Syntax and semantics aren't separable in the way that we've been assuming
- Grammars contain form-meaning pairings that vary in the degree to which the meaning of a constituent (and what constitutes a constituent) can be computed from the meanings of the parts.

Constructional Approach

- So we'll allow both

  VP → V NP {V.sem(NP.sem)}

  and

  VP → Kick-Verb the bucket {λ x Die(x)}

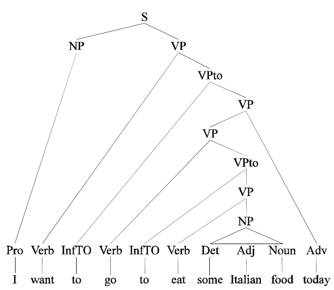
Computational Realizations

- Semantic grammars
  - Simple idea, dumb name
- Cascaded finite-state transducers
  - Just like Chapter 3

Semantic Grammars

- One problem with traditional 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

BERP Example



BERP Example

- How about a rule like the following…

```
Request → I want to go to eat FoodType Time
                    { some attachment }
```


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

Semantic Grammars

- Typically used in conversational agents in constrained domains
  - Limited vocabulary
  - Limited grammatical complexity
  - Chart parsing (Earley) can often produce all that's needed for semantic interpretation even in the face of ungrammatical input.