Semantic Processing

L445 / L545

Dept. of Linguistics, Indiana University

Spring 2017

Semantics

- **Semantics** = study of meaning
  - We want to investigate the literal meaning of sentences → **compositional semantics**
  - Lexical semantics = study of meaning of words
    - Word Sense Disambiguation deals with lexical semantics
- To represent the meaning of a sentence, we choose First-Order Predicate Calculus (FOPC) & a basic (Davidsonian) event semantics
  - *I have a car*
  - $\exists x, y$ Having (x) ∧ Haver(Speaker,x) ∧ ThingHad(y,x) ∧ Car(y)

Part I: Semantic Representation

There are a variety of way to represent semantics, all sharing some commonalities:

- **Unambiguous** representation: the underlying semantic representation of a sentence should be unambiguous
  - A sentence might mean multiple things
  - But each meaning is represented unambiguously
- Allows for **vagueness**: a semantic representation can be partly undefined
  - *I eat Italian food.*
  - Not clear exactly what *Italian food* refers to.
- **Verifiable**: is a particular sentence true or false?

See also Blackburn and Bos (2003), http://www.let.rug.nl/bos/comsem/book1.html

Canonical Form

Furthermore, if two distinct sentences mean the same thing, they should have the same semantic representation.

- The **canonical form** is the semantic form for all sentences with the same semantics

(1) a. Does Maharani have vegetarian dishes?
  b. Do they have vegetarian food at Maharani?
  c. Are vegetarian dishes served at Maharani?
  d. Does Maharani serve vegetarian food?

- All of these sentences should probably have the same representation (for many purposes)

Model-Theoretic Semantics

Semantic representations are formalized with a **model**

- A model represents the state of affairs in the world being represented
  1. Represent objects, properties of objects, & relations between them
  2. Successfully map the meaning representation to the world being considered

Meaning representation:

- Non-logical vocabulary: names of objects, properties, & relations
  - Denotation: every element of non-logical vocab corresponds to a fixed, well-defined part of model
- Logical vocabulary: closed set of symbols, operators, quanitifers, links, etc.: needed to compose expressions

Denotation

Extensional approach to meaning: denotation is reducible to sets

- Domain: set of objects/elements that are part of state of affairs
- Properties: sets of domain elements which have property in question
- Relations: sets of ordered lists/tuples of domain elements

Interpretation: Mapping from meaning representations to denotation

Model of restaurant world

- Domain: $\mathcal{D} = \{a, b, c, d, e, f, g, h, i, j\}$
  - Matthew, Franco, Katie, & Caroline: a, b, c, d
  - Frasca, Med, Rio: e, f, g
- Properties
  - Frasca, Med, and Rio are noisy: $Noisy = \{e, f, g\}$
- Relations
  - Matthew likes the Med.
  - Katie likes the Med and Rio.
  - $Likes = \{< a, f >, < c, f >, < c, g >\}$

Predicate-Argument Structure

Recall verb subcategorization requirement

- We can **link** these syntactic argument slots with semantic roles, or **thematic (theta) roles**

Syntactic role	Semantic role
Subject NP	Agent
Object NP	Patient

- We can further restrict such theta roles to meet certain conditions, so-called **selectional restrictions**
  - e.g., the agent role of *eat* must be an animal

Towards a Representation

We can represent verbs with semantic roles by:

- defining a semantic predicate for that verb (e.g. *Eat*)
- giving the predicate the appropriate number of slots (e.g., 2)

$$\text{NP}_x \textit{ eats } \text{NP}_y \Rightarrow Eat(x, y)$$

- The slots are filled in by **variables** (e.g., x, y), until we can fill them by actual information from a sentence

Now to define the structures that are allowed ...

First-Order Predicate Calculus (FOPC)

Predicates:

- Predicates take arguments & define the relation among them, e.g. *Eat* takes two arguments (eater/eaten)

Terms, or devices to represent objects:

- **Constants**: specific objects in the world e.g., *John* and *fruit* in $Eat(John, fruit)$
- **Variables**: like constants, but not totally specified e.g., x in $Eat(John, x)$ $\rightarrow$: no specification of what John eats
- **Functions**: refer to unique objects which are complex e.g., *the restaurant's location* becomes $LocationOf(Restaurant)$

Why FOPC?

Advantages of first-order predicate calculus (FOPC):

- Proving FOPC statements is efficient
- FOPC statements can be linked to syntactic rules
- FOPC deals with a wide range of linguistic phenomena

Logical Connectives

We can build up predicates and then combine them with logical connectives

- **not** ($\neg$): I am not happy: $\neg Happy(Speaker)$
- **and** ($\wedge$): I am happy and free: $Happy(Speaker) \wedge Free(Speaker)$
- **or** ($\vee$): I am happy or I'm free: $Happy(Speaker) \vee Free(Speaker)$
  - This is an inclusive *or*: it is true if the speaker is both happy and free (as we'll see momentarily)
- **if** ($\Rightarrow$): If I'm free, then I'm happy: $Free(Speaker) \Rightarrow Happy(Speaker)$

Variables and Quantifiers

Variables allow a slot to be unfilled, but we need to **quantify** over (restrict) such variables

- 'there exists' (∃): a restaurant that serves Mexican food: $\exists x Restaurant(x) \wedge Serves(x, MexicanFood)$
  - Substituting a single restaurant which serves Mexican food for x will make this logical formula true
- 'for all' (∀): All vegetarian restaurants serve vegetarian food: $\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$
  - For this to be true, all substitutions for x that make VegetarianRestaurant(x) true must also make Serves(x,VegetarianFood) true

Determining Truth

- Truth-conditional semantics: sentences are analyzed in terms of whether or not they evaluate to true, with respect to some model

To determine whether something is true or not, we evaluate each predicate to see if it's true, and the connectives are interpreted as follows (T=True, F=False):

p	q	¬p	p∧q	p∨q	p⇒q
F	F	T	F	F	T
F	T	T	F	T	T
T	F	F	F	T	F
T	T	F	T	T	T

- Possible-worlds semantics: same idea, but true for a given "possible world"

Rules of Inference

Rules of **inference** allow us to draw conclusions based on what information we have

- Can add information to database of information

Modus ponens: two statements combine to make a third true:

- All men are mortal ($\forall x[man(x) \rightarrow mortal(x)]$)
- Socrates is a man ($man(Socrates)$)
- Therefore, Socrates is mortal ($mortal(Socrates)$)

Forward/backward chaining

Forward chaining (as in production systems)

- Add individual facts to the knowledge base & use modus ponens to fire implications
- New facts can then cause modus ponens to fire again
- All inference is performed in advance

Backward chaining

- Modus ponens is run in reverse to prove queries
- If query proposition is not in the knowledge base, try to prove it
  - We don't know if $Serves(Leaf, VegetarianFood)$
  - But we know: $VegetarianRestaurant(Leaf)$ and $VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

Back to language: Meaning Postulates

What's wrong with a representation like $Eats(John, Fruit)$?

- Is it the same event as $Eats_2(John, Fruit, Table)$ (where *John eats fruit at the table*)?
- Could make a meaning postulate:
  - $\forall x, y, z Eats(x, y, z) \Rightarrow_{MP} Eats(x, y)$

Meaning postulates can generally be used to relate, e.g., *Eating* and *Hunger*, but it seems unsatisfactory here

Representing Events

A representation like $Eats(John, Fruit)$ and its subsequent meaning postulates can be kind of messy:

- We will instead treat the eating event as a variable:
  - $Isa(w, Eating)$ (w is an ("isa") *Eating* event)
  - Actually: there is a w such that this is true: $\exists w Isa(w, Eating)$
- Each argument is then given its own predicate: $Eater(w, John)$, $Eaten(w, Fruit)$
- Combine them with connectives: $\exists w Isa(w, Eating) \wedge Eater(w, John) \wedge Eaten(w, Fruit)$

This allows us to easily modify these events, e.g., $Location(w, RuncibleSpoon)$

Representing Time

New predicates represent time/tense information, to relate sentences to the present moment:

(2) a. I arrive in Peoria

b. $\exists i, e, w, t\ Isa(w, Arriving) \wedge Arriver(w, Speaker) \wedge Destination(w, Peoria)$

(3) a. I arrived in Peoria: ... $\wedge Interval(w, i) \wedge EndPoint(i, e) \wedge Precedes(e, Now)$

b. I am arriving in Peoria: ... $\wedge Interval(w, i) \wedge MemberOf(i, Now)$

c. I will arrive in Peoria: ... $\wedge Interval(w, i) \wedge EndPoint(i, e) \wedge Precedes(Now, e)$

More constructions

Can augment semantic representations to handle:

- Verbal aspect: *I live in Bloomington* vs. *I am living in Bloomington*
- Belief: *I believe unicorns exist* doesn't make *Unicorns exist* true
- Modals: semantic contribution of *may*, *must*, etc.

Shortcomings of FOPC by itself

There's often a difficulty in figuring out what logical connectives are involved

- *if* statements that don't mean *if*

(4) a. If you're interested in baseball, the Rockies are playing tonight.

b. ?? $HaveInterestIn(Hearer, Baseball) \Rightarrow Playing(Rockies, Tonight)$

- *and* statements that do mean *if*

(5) a. One more beer, and I'll fall off this stool

b. ?? *Beer*... $\wedge$ *Fall*...

Furthermore, constants like *VegetarianFood* have no relation to constants like *VegetarianRestaurant*

Description Logics

An alternate representation

Semantic networks: objects are nodes in a graph, and relations are named links between objects

- **Description logics** specify the semantics of structured network representations

Emphasize representation of knowledge about categories, individuals belonging to those categories, & relationships among individuals

- Terminology: set of concepts making up a domain
- TBox: portion of knowledge base containing terminology
- ABox: portion of knowledge base containing facts about individuals
- Ontology: captures subset/superset relations among categories

Subsumption

To specify hierarchy, we assert **subsumption** relations

- Restaurant $\sqsubseteq$ CommercialEstablishment
- ItalianRestaurant $\sqsubseteq$ Restaurant
- ChineseRestaurant $\sqsubseteq$ Restaurant

Formally, these are interpreted as subset relations

- Can a restaurant be both Italian and Chinese?
  - Specify disjointness: ChineseRestaurant $\sqsubseteq$ **not** ItalianRestaurant
  - Fully cover a category: Restaurant $\sqsubseteq$ (**or** ItalianRestaurant ChineseRestaurant MexicanRestaurant)

Relations

Relations (or roles/role-relations) specify what it means to be a member of a category

- ItalianCuisine $\sqsubseteq$ Cuisine
- ItalianRestaurant $\sqsubseteq$ Restaurant $\sqcap$ $\exists$ hasCuisine.ItalianCuisine

Read as: 'Individuals in the ItalianRestaurant category are subsumed by Restaurant category and an unnamed class: set of entities serving Italian cuisine'

- Existential clause defines unnamed class
- Equivalent FOL: $\forall x ItalianRestaurant(x) \rightarrow Restaurant(x) \wedge (\exists y Serves(x, y) \wedge ItalianCuisine(y))$

Inference
Subsumption

Based on the facts in a terminology, subsumption checks if a superset/subset relation exists between 2 concepts

Assume that we have *defined* Italian Restaurants as follows:

- ItalianRestaurant $\sqsubseteq$ Restaurant $\sqcap$ $\exists$ hasCuisine.ItalianCuisine

and we then add this fact:

- IlFornaio $\sqsubseteq$ ModerateRestaurant $\sqcap$ $\exists$ Cuisine.ItalianCuisine

Subsumption checks whether the following fact is true:

- IlFornaio $\sqsubseteq$ ItalianRestaurant
  - ModerateRestaurant $\sqsubseteq$ Restaurant
  - $\exists$ Cuisine.ItalianCuisine restriction is met

Inference
Instance checking

Instance checking: determining whether an individual can be classified as a member of a particular category

- Compare known relations & categorical statements to current knowledge
- Return a list of the most specific categories it belongs to

New facts about the individual Gondolier:

- Restaurant(Gondolier)
- hasCuisine(Gondolier, ItalianCuisine)

Can now try to determine if Gondolier is Italian, vegetarian, has moderate prices, etc.

Part II: Deriving a Semantic Analysis

We will focus on two main ways of analyzing the semantics of a sentence:

- Syntax-driven semantic analysis: build up a semantic parse alongside a syntactic parse
  - Requires that we have a semantic form associated with every lexical item and every rule
- Semantic grammars: a more robust way to extract semantic information
  - Not every word will have a semantic form, but we'll be able to find what we want to find

Principle of Compositionality

The meaning of a sentence is composed of the meaning of its parts

- The way we syntactically compose a sentence determines how we semantically compose it
- For every syntactic rule, there is a corresponding semantic rule (rule-to-rule hypothesis)

Semantic analyzer: take the output of a parser & figure out the meaning

Augmenting Context-free Rules

Augment context-free rules with semantic attachments

Lexical items (first pass):

- MassNoun $\rightarrow$ meat $\{Meat\}$
- Verb $\rightarrow$ serves $\{\exists e, x, y\ Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, y)\}$

Rules:

- NP $\rightarrow$ MassNoun $\{MassNoun.sem\}$
- VP $\rightarrow$ Verb NP $\{Verb.sem(NP.sem)\}$

Tree Structure

For the phrase *serves meat*:

VP:??

V:$\exists e, x, y\ Isa(e, Serving)$... NP:*Meat*

MassN:*Meat*

From existential to instantiated

We would like the semantic value of the VP to be: $\exists e, x$ $Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, Meat)$

But how do we go

- from: $\exists e, x, y$ $Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, y)$
- to: $\exists e, x$ $Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, Meat)$

i.e., from "there is a y" to instantiating y as *Meat*

Lambdas (Currying)

Instead of saying "there exists a y", what we want to say is: we have a value of y which is waiting to be filled in.

- A λ (lambda) will do this for us
  - **Currying** a predicate with multiple arguments into single argument predicates
- $\lambda x P(x)$ means that x will be replaced by something else, which will then be an argument of P

This is how we apply so-called λ-reduction:

- $\lambda x P(x)(A)$
- $P(A)$

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

A revised lexical entry for *serves*

- Verb → serves $\{\lambda y \lambda x \exists e$ $Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, y)\}$
  1. Take an argument for y & put it into the *Served* relation
  2. Take an argument for x & put it into the *Server* relation

For VP → Verb NP $\{Verb.sem(NP.sem)\}$, with $NP.sem = Meat$, we have the following:

- $\lambda y[\lambda x \exists e$ $Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, y)](Meat)$
- $\lambda x \exists e\ Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, Meat)$

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

What about quantifiers?

How do we handle NPs like *a restaurant*?

- Det → a $\{\exists\}$ [or a more elaborate semantics]
- Noun → restaurant $\{Restaurant\}$
- Nominal → Noun $\{\lambda x\ Isa(x, Noun.sem)\}$
- NP → Det Nominal $\{Det.sem\ x\ Nominal.sem(x)\}$

The resulting meaning representation will be: $\exists x$ $Isa(x, Restaurant)$

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

Semantic Problem #1

Quantifier scoping

One major problem we are (for the most part) ignoring is that of quantifier scoping

(6) Every student likes some book

- $\forall x\ [Student(x) \Rightarrow \exists y\ [book(y) \wedge like(x, y)]]$
- $\exists y\ [book(y) \wedge \forall x\ [Student(x) \Rightarrow like(x, y)]]$

Some solutions for determining quantifier scope:

- Quantifier storage: store quantifiers in the tree until you need them
- Semantic underspecification of scope
- Scope heuristics (left-to-right; domain-specific heuristics; etc.)

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

Store and Retrieve Approaches

First, we need *underspecified representations* that embody all readings without enumerating all of them

Cooper storage:

- Replace single semantic attachments with a store
  - Core meaning representation
  - Indexed list of quantified expressions gathered from nodes below this one
    - λ-expressions that combine with core meaning to incorporate quantifiers in the right way

Top node of a parse tree for *Every restaurant has a menu*:

$$\exists e\ Having(e) \wedge Haver(e, s_1) \wedge Haved(e, s_2)$$
$$(\lambda Q.\forall x Restaurant(x) \Rightarrow Q(x), 1),$$
$$(\lambda Q.\exists x Menu(x) \wedge Q(x), 2)$$

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

Store and Retrieve Approaches (2)

$$\exists e\; Having(e) \land Haver(e, s_1) \land Haved(e, s_2)$$
$$(\lambda Q.\forall x Restaurant(x) \Rightarrow Q(x), 1),$$
$$(\lambda Q.\exists x Menu(x) \land Q(x), 2)$$

To get two different meanings, pull the quantifiers out of storage in different orders

To make all this work, we also have to adjust the syntactic/semantic rules

- NPs combining a Det and Nom introduce a λ around an indexed variable
- Determiner semantics gets put into storage

Hole semantics

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

A different approach to underspecifying meaning is that of **hole semantics**

- λ-variables are replaced with *holes*
- All FOL subexpressions are given *labels*
  - *dominance constraints* restrict which labels can fill which holes
  - e.g., $l \leq h$: expression containing hole h dominates expression with label l

Every restaurant has a menu:

$$l_1 : \forall x Restaurant(x) \Rightarrow h_1$$
$$l_2 : \exists y Menu(y) \land h_2$$
$$l_3 : \exists e Having(e) \land Haver(e, x) \land Had(e, y)$$
$$l_1 \leq h_0, l_2 \leq h_0, l_3 \leq h_1, l_3 \leq h_2$$

Hole semantics (2)

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

$$l_1 : \forall x Restaurant(x) \Rightarrow h_1$$
$$l_2 : \exists y Menu(y) \land h_2$$
$$l_3 : \exists e Having(e) \land Haver(e, x) \land Had(e, y)$$
$$l_1 \leq h_0, l_2 \leq h_0, l_3 \leq h_1, l_3 \leq h_2$$

Now, need a **plugging** method to fill the holes

- Can fill h_0 with either l_1 or l_2 as h_0 dominates both and neither one dominates the other
- e.g., $P(h_0) = l_1$, which then leads to $P(h_1) = l_2$ and $P(h_2) = l_3$

Advantages of hole semantics

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

1. Not dependent upon any particular grammatical construction (e.g., NPs)
   - Can label or designate as holes any aribtrary FOL formula
2. Dominance constraints can rule out unwanted constraints, but without fully specifying the meaning
   - Constraints can come from specific lexical & syntactic knowledge

Semantic Problem #2

Intersecting vs. Scoping Adjectives

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

Consider the following:

(7) cheap restaurant: $\lambda x\; Isa(x, Restaurant) \land Isa(x, Cheap)$

(8) a. small elephant → an elephant is not a small thing (only in relation to other elephants)

b. fake gun → a *fake gun* is not a gun

- *cheap restaurant* is **intersective**, simply intersecting the semantics of *cheap* with *restaurant*
- *small elephant* is sort of intersective, but *small* has to be interpreted w.r.t. a context
- *fake gun* involves an adjective which scopes over the noun, so its semantics should resemble a verb's: $Fake(Gun(x))$

Parsing with Semantic Constraints

Semantic Processing
Semantic Representation
FOPC
Inference
Issues
Description Logics
Semantic Processing
Augmenting CFGs
Currying
Quantifier scope
Semantic Grammars

Can use our semantic information to restrict our parses, e.g., in an Earley parser

(9) # The tree ate my dinner.

Alter the Earley algorithm:

- Keep a field for semantic attachments
- Unify syntactic trees, if able
- Compute semantic analysis and note if it is a valid meaning representation (or perhaps conflicts with what is in the information database)

Semantic Grammars

Instead of mapping semantic rules to syntactic rules, we could just write semantic rules instead.

- *Nominal* → *AdjNominal* is split up into rules like *FoodType* → *Nationality FoodType*
- This becomes close to template filling: *InfoRequest* → *when does Flight arrive in City*

Advantages:

- Previous example will work even with a sentence like *When does it arrive in Dallas?*
- Avoid dealing with syntactic constituents that have virtually no meaning or add vacuous meaning

Disadvantages of Semantic Grammars

- Not easily reusable ... e.g., have to be talking about flights
- Have a huge explosion of rules
  - *vegetarian restaurant*, *California restaurant*, *expensive restaurant*, and *pasta restaurant* all need different entries
- Doesn't match linguistic theory, or intuitions about what happens with language processing

Typically work best in restricted domains