6.863J Natural Language Processing
Lecture 17: Scoping outthe meaning of it all, #3
(or #42)

Instructor: Robert C. Berwick
berwick@ai.mit.edu

The Menu Bar

- Administrivia:
  - Lab 4a out April 14

Agenda:

The lambda calculus picture

Today: beyond simple VPs; properties & database interfaces; adjectives, prepositions

Scoping ambiguities & computation

Meaning of meaning, redux

- How can we automate process of associating semantic representations w/ expressions of natural language?
- How can we use semantic representations to automate drawing inferences?

Why compositionality?

- a human being can understand a possibly *infinite* number of sentences never heard before (namely by constructing their meaning from a *finite* set of rules and a *finite* set of known lexical meanings).
- Also, a compositional account of meaning suggests a plausible explanation of why we perceive a connection *in meaning* between sentences that share syntactic parts

The recipe

The lexical items (i.e. the words) in a sentence give us the basic ingredients for our representation.

Syntactic structure tells us how the semantic contributions of the parts of a sentence are to be joined together.

The three tasks

- Specify a reasonable syntax for the natural language fragment of interest.
- Specify semantic representations for the lexical items.
- Specify the translation of complex expressions (i.e. phrases and sentences) compositionally - That is, we need to *systematically* specify the translation of such expressions in terms of the translation of their parts, "parts" here referring to the substructure given to us by the syntax.

Our working vocabulary

Formula	sequence/tree of symbols	$x, y, f, g, p, 1, \pi, \in, \neg, \wedge, \forall, \exists$
Model	something we understand	natural numbers or sets
Interpretation	maps formulae into models	$[\![\text{three plus five}]\!] = 8$
logical consequence	$A \models B$, iff $\mathcal{M} \models B$ for all $\mathcal{M} \models A$	

Lambda calculus & lambda terms

- Key to building logical semantic representations
- Extension of FOPC allowing us to bind variables using operator λ
- Occurrences of variables bound by λ are placedholders for missing information: they explicitly mark where we should substitute the pieces we obtain during semantic construction
- Like a programming language devoted to task of gluing the items together for semantic rep – a construction kit, it's the Elmer's glue

The lambda operator

- The lambda operator marks missing information by binding variables
- E.g., a lambda expression:
  λx.person(x)
- The prefix λx binds the occurrence of x in person(x)
- λx.person(x) can be read as: "I am the 1-place predicate and I'm looking for a term to fill my argument slot"

Lambda calculus

- Glue language – a 'programming language' with a single task: gluing together items to build semantic representations
- Way of controlling substitutions
- Instructions – β and α conversion
- Functional application: β conversion
  λx.person(x)@bob
- The expression λx.person(x) is the functor
- The expression bob is the argument
- The @ operator indicates functional application = a substitution
- Fill the vars in the functor by occurrences of arg bob

β– conversion does the work

- Actual substitution is performed by β– conversion:

From:

λx.person(x)@bob

- β– conversion produces:

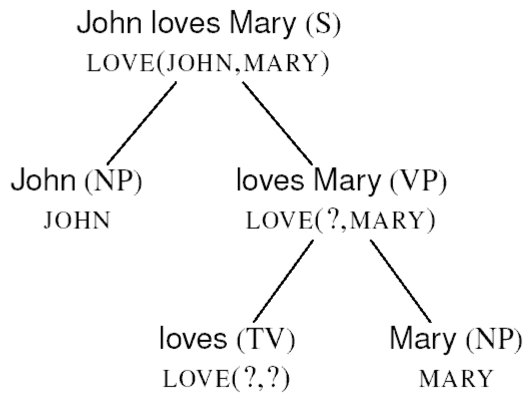
person(bob)

- Throw away the λx at the start, substitute the argument for all occurrences of x bound by λx

What are λ terms?

- They are functions
  - λ x.P(x) : function from objects to truth values (ind to Bool)
  - λ P.P(john) : function from predicates to truth values
  - λ P. λ x.P(x) : function from predicates to functions from objects to truth values…

Filling the semantic gaps…



A woman walks

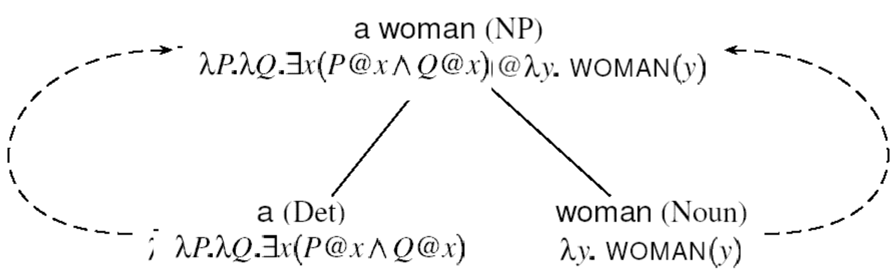
- First order formula $\exists x\ (WOMAN(x) \wedge WALK(x))$
- The verb 'walks' contributes the predicate symbol WALK
- The determiner must contribute the quantifier *and* the pattern of the quantification
- What's the lexical template pattern? Abstract away the variables x

‘a woman walks’ - general

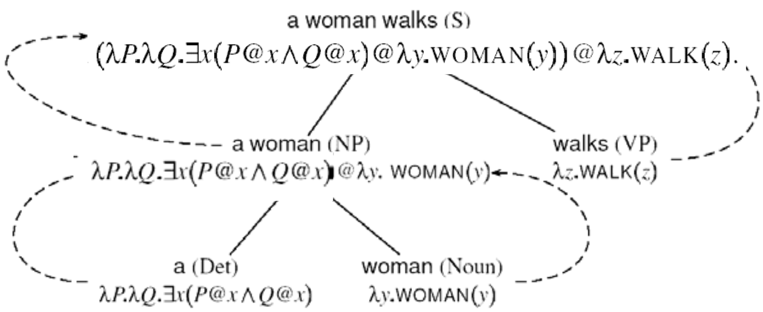
$\lambda P.\lambda Q.(\exists(P@x \wedge Q@x))$

- P and Q stand for missing predicate symbols

A woman



The whole S



Beta reductions

$$(\lambda P.\lambda Q.\exists x(P@x \wedge Q@x)@\lambda y.\mathrm{WOMAN}(y))@\lambda z.\mathrm{WALK}(z).$$

$$\lambda Q.\exists x(\lambda y.\mathrm{WOMAN}(y)@x \wedge Q@x)@\lambda z.\mathrm{WALK}(z)$$

$$\exists x(\lambda y.\mathrm{WOMAN}(y)@x \wedge \lambda z.\mathrm{WALK}(z)@x)$$

$$\exists x(\mathrm{WOMAN}(x) \wedge \mathrm{WALKS}(x))$$

The whole S – goal is love(bob, ice-cream)

Bob loves ice-cream (S)

(λP P. BOB) λX λz (X. λx LOVE(z,x)) (λP P.ICE-CREAM)

Bob (NP)

λP P.BOB

loves ice-cream (VP)

λX λz (X. λx LOVE(z,x))

loves (trans verb)

λX λz (X. λx LOVE(z,x))

ice-cream (NP)

λP P. ICE-CREAM

Lambda reduction

Start with:

(λP P. BOB) (λ**X** λz (**<u>X</u>**. λx LOVE(z,x)) (λP P.ICE-CREAM))

↓

(λP P. BOB) (λz (λ**P** **<u>P</u>**.ICE-CREAM). λx LOVE(z,x))

↓

(λP P. BOB) (λz (λ**x** LOVE(z,**<u>x</u>**).ICE-CREAM))

↓

(λ**P** **<u>P</u>**. BOB) (λz LOVE(z,ice-cream))

↓

(λ**z** LOVE(**<u>z</u>**,ice-cream). BOB)

↓

LOVE(BOB,ice-cream)

Now we can…uhm, go to town…

- Adjectives are properties

red: Adj: $\lambda P\,(\lambda x\, P(x) \wedge red'(x))$

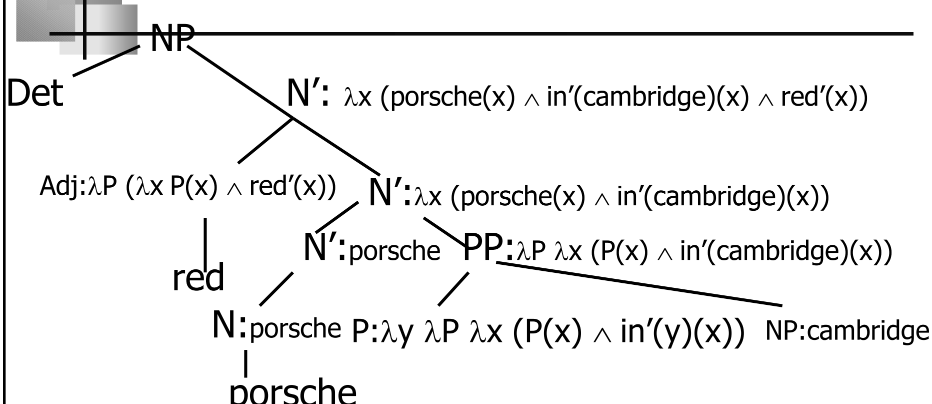
- Prepositions are properties

in: P: $\lambda y\, \lambda P\, \lambda x\, (P(x) \wedge in'(y)(x))$

- Everything is a property…!

- …red Porsche in cambridge…

"…red porsche in cambridge"



What's the parse?

Red (porsche in c.)

Red porsche in cambridge

- Isn't there another parse?
- Yes: (red porsche) in cambridge

- Why doesn't this matter?

Translation to SQL – property semantics

- **`Select Porsches.obj from Porsches, locations, Red where Porsches.obj = Locations.obj AND Locations.place = 'cambridge' AND Porsches.obj = Red.obj`**

- Now – let's add 'some', as in "some red Porsche in cambridge"
- Some = $\lambda x.P(x)$; Some(P)=yes iff $\exists x P(x)$=yes, i.e., if $P \neq \emptyset$

Translation to SQL

- **Select Porsche.obj from Porsches, locations, Red where Porsches.obj = Locations.obj AND Locations.place = 'cambridge' AND Porsches.obj = Red.obj <u>having count(*)> 0</u>**

More on adjectives

- What's the difference?
- Red Porsche in Cambridge
- Fake Porsche in Cambridge

- Is there a difference in parse now?

Question answering

- Yes/no question: interrogate DB (not just assert or check the fact as with a statement)
- Content question: Who loves ice-cream? is an open proposition – which individuals make statement true

- Need semantics for who, what, which, how_many

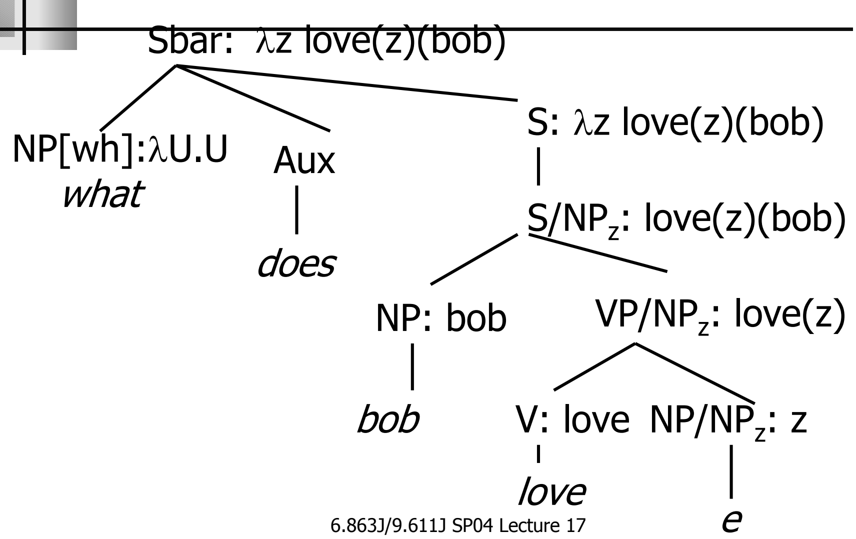
Question answering

- What, NP[wh]: λ U. U
- Who, NP[wh]: λ U λ x U(x) $\wedge$ human(x)
- Which, Det[wh]: λ P λ U λ x P(x) $\wedge$ U(x) (why do we have the U??)
- How_many, Det[wh]: λ P λ U |λ x P(x) $\wedge$ U(x) |

Question sentence – new rules

- Sbar: NP[wh] Aux S
- Filler: S $\rightarrow$ S-NP_z
- Gaps: NP-NP_z : z $\rightarrow$ e
- and need a way to link fillers and gaps:
- S: λz F(…z…) $\rightarrow$ S-NP_z : F(…z…)

What does Bob love *e*



SQL

- **`Select loved from Loves where Loves.lover = 'bob'`**

- *Who does Bob love*
- *Which cars does Bob love*

  ```
  select liked from Cars.Likes where
  Cars.obj = Likes.liked AND
  Likes.liker = 'bob'
  ```
- *Which cars does every student love*

Why λ calculus works

The process of combining two representations was perfectly uniform. We simply said which of the representations is the functor and which the argument, whereupon combination could be carried out by applying functor to argument and β-converting. We didn't have to make any complicated considerations here.

The load of semantic analysis was carried by the lexicon: We used the λ-calculus to make missing information stipulations when we gave the meanings of the *words* in our sentences. For this task, we had to think accurately. But we could make our stipulations declaratively, without hacking them into the combination process.

What you must do

We have to locate gaps to be abstracted over in the partial formula for our lexical item. In other words, we have to decide *where to put* the λ-bound variables inside our abstraction. For example when giving the representation $\lambda P.P@\text{MARY}$ for the proper name 'Mary' we decided to stipulate a missing functor. Thus we applied a λ-abstracted variable to MARY.

We have to decide *how to arrange* the λ-prefixes. This is how we control in which order the arguments have to be supplied so that they end up in the right places after β-reduction when our abstraction is applied. For example we chose the order $\lambda P.\lambda Q$ when we gave the representation $\lambda P.\lambda Q.\exists x(P@x \wedge Q@x)$ for the indefinite determiner 'a'. This means that we will first have to supply it with the argument for the restriction of the determiner, and then with the one for the scope.

Accidental bindings (alpha reduction)

Before we can put λ-calculus to use in an implementation, we still have to deal with one rather technical point: Sometimes we have to pay a little bit of attention which variable names we use. Suppose that the expression $\mathcal{F}$ in $\lambda V.\mathcal{F}$ is a complex expression containing many λ operators. Now, it could happen that when we apply a functor $\lambda V.\mathcal{F}$ to an argument $\mathcal{A}$, some occurrence of a variable that is is free in $\mathcal{A}$ becomes bound when we substitute it into $\mathcal{F}$.

Example

For example when we construct the semantic representation for the verb phrase 'loves a woman', syntactic analysis of the phrase could lead to the representation:

$$\lambda P.\lambda y.(P@\lambda x.\text{LOVE}(y,x))@(\lambda Q.\lambda R.(\exists y(Q@(y)\wedge R@y))@\lambda w.\text{WOMAN}(w))$$

β-reducing three times yields:

$$\lambda y.(\lambda R.(\exists y(\text{WOMAN}(y)\wedge R@y))@\lambda x.\text{LOVE}(y,x))$$

Notice that the variable y occurs λ-bound as well as existentially bound in this expression. In $\text{LOVE}(y,x)$ it is bound by λy, while in $\text{WOMAN}(y)$ and R it is bound by $\exists y$.

Example – w/ problem after 1 more beta reduction

$$\lambda y.(\exists y(\text{WOMAN}(y)\wedge\lambda x.\text{LOVE}(y,x)@y))$$

$\text{LOVE}(y,x)$ has been moved inside the scope of $\exists y$. In result, the occurence of y has been 'caught' by the existential quantifier, and λy doesn't bind any occurence of a variable at all any more. Now we β-convert one last time and get:

$$\lambda y.(\exists y(\text{WOMAN}(y)\wedge\text{LOVE}(y,y)))$$

Must rename variables = α reduction

So

- Most work done in the lexicon
- Pattern for intransitive, transitive verbs – can be abstracted by a ‘macro’ (eg, ‘sleep’ is like ‘walk’…)
- What about a ditransitive verb? (eg, give?)

Puzzles: Unknown vocabulary problem

- Everyone is a therapist, but Cinderella isn't
- What will a logic system do?

Still some puzzles…!!

- Some person broke every Porsche

Semantic ambiguity

- Every student did not pass the exam (One exam or many?)

Quantifier ambiguity

1. $\forall x.\text{STUDENT}(x) \rightarrow \neg\text{PASS}(x)$
2. $\neg\forall x.(\text{STUDENT}(x) \rightarrow \text{PASS}(x))$

1. Negation has <u>narrow scope</u>, det has <u>wide scope</u>
2. Negation has <u>wide scope</u>, det has <u>narrow scope</u>

Are we done yet?

- Every owner of a siamese cat loves a therapist
- Three quantifiers – so, if they freely permute
- Five readings
- But… some are logically equivalent! Only 5…

Cats

$\exists z.\text{THERAPIST}(z) \wedge \exists y.\text{SIAMESECAT}(y) \wedge \forall x.((\text{OWNER}(x) \wedge \text{OF}(y,x)) \rightarrow \text{LOVE}(x,z))$

$(\mathsf{A}@\mathsf{therapist})@\lambda z.[(\mathsf{A}@\mathsf{s\_cat})\lambda y.[(\mathsf{Every}@\lambda x.[\text{OWNER}(x) \wedge \text{OF}(y,x)])@\lambda x.\text{LOVE}(x,z)]]$

More examples

- Possibly a dog is barking
- Adverbs interact!

- But our algorithm so far is <u>determinstic</u> – we get only the det wide scope reading
- Why?
- Look at the reduction enterprise…

Reduction order and scope – 'every man loves a woman'

- Let EVERY stand for

$$\lambda P \lambda Q \forall x(P(x) \rightarrow Q(x))$$

- Let A stand for the existential. Then:

$$(\text{Every}@\lambda v.\text{MAN}(v))@((\lambda P \lambda x P@(\lambda y.\text{LOVE}(x,y)))@(A@\lambda w.\text{WOMAN}(w)))$$

- after beta reduction 1:

$$(\text{Every}@\lambda v.\text{MAN}(v))@(\lambda x(A@\lambda w.\text{WOMAN}(w))@(\lambda y.\text{LOVE}(x,y)))$$

What is a solution?

- Montague: let's raise the existential to this:

$$(A@\lambda w.\text{WOMAN}(w))@(\lambda y.(\text{Every}@\lambda v.\text{MAN}(v))@(\lambda x.\text{LOVE}(x,y)))$$

- Does this work? Why did we drag along the λ y?

Comparison

$$(\text{Every}@\lambda v.\text{MAN}(v))@(\lambda x.(\text{A}@\lambda w.\text{WOMAN}(w))@(\lambda y.\text{LOVE}(x,y)))$$

$$(\text{A}@\lambda w.\text{WOMAN}(w))@(\lambda y.(\text{Every}@\lambda v.\text{MAN}(v))@(\lambda x.\text{LOVE}(x,y)))$$

Montague’s solution (and others’ solution…)

- Two different syntactic analyses…
- One, the familiar one
- The second: ‘A woman, every man loves her’
- This is quantifier raising (in syntax – R. May, 1977) or “quantifying in” (Montague’s phrase)
- Let’s see how it works
- Consider semantics for “Every man loves her”
- “her” = λ P.P(v1)

Every man loves her

$$(\lambda P \lambda Q \forall x.(P(x) \rightarrow Q(x)) @ \lambda y.\text{MAN}(y)) @ (\lambda R \lambda x R @ \lambda y.\text{LOVE}(x, y) @ \lambda P.P(v_1))$$

Beta reduced to this…

$$\forall x(\text{MAN}(x) \rightarrow \text{LOVE}(x, v_1))$$

Now we want to put the "a woman" in front

Getting the scope right

- We have to <u>delay</u> processing NP 'a woman' until we have processed NP 'every man'
- This lifts the existential quantifier above the universal

This formula

$$\underline{\lambda Q \exists y.(\mathrm{WOMAN}(y) \wedge Q(y))} @ (\lambda v_1.\forall x(\mathrm{MAN}(x) \rightarrow \mathrm{LOVE}(x, v_1)))$$

- Beta reduces to:

$$\exists y.(\mathrm{WOMAN}(y) \wedge \forall x(\mathrm{MAN}(x) \rightarrow \mathrm{LOVE}(x, y)))$$

Cats

$$\exists z.\mathrm{THERAPIST}(z) \wedge \exists y.\mathrm{SIAMESECAT}(y) \wedge \forall x.((\mathrm{OWNER}(x) \wedge \mathrm{OF}(y,x)) \rightarrow \mathrm{LOVE}(x,z))$$

$$(\mathsf{A}@\mathsf{therapist})@\lambda \mathsf{z}.[(\mathsf{A}@\mathsf{s\_cat})\lambda \mathsf{y}.[(\mathsf{Every}@\lambda \mathsf{x}.[\mathrm{OWNER}(\mathsf{x}) \wedge \mathrm{OF}(\mathsf{y},\mathsf{x})])@\lambda \mathsf{x}.\mathrm{LOVE}(\mathsf{x},\mathsf{z})]]$$

Cats…not!

$\exists z.\text{THERAPIST}(z) \wedge \exists y.\text{SIAMESECAT}(y) \wedge \forall x.((\text{OWNER}(x) \wedge \text{OF}(y,x)) \rightarrow \text{LOVE}(x,z))$

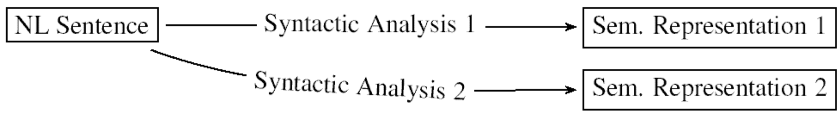
$(\text{A@therapist})@\lambda z.[(\text{A@s\_cat})\lambda y.[(\text{Every}@\lambda x.[\text{OWNER}(x) \wedge \text{OF}(y,x)])@\lambda x.\text{LOVE}(x,z)]]$

$*\forall x.\text{OWNER}(x) \wedge \text{OF}(y,x) \wedge \exists y.\text{SIAMESECAT}(y) \rightarrow \exists z.\text{THERAPIST}(z) \wedge \text{LOVE}(x,z)$

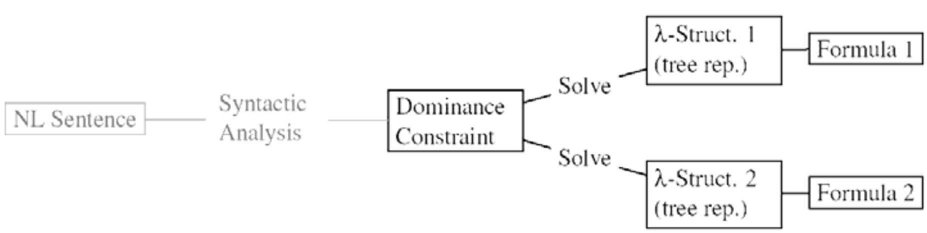
readings

number of quantifers	readings
4	14
5	42
6	132
7	429
8	1430

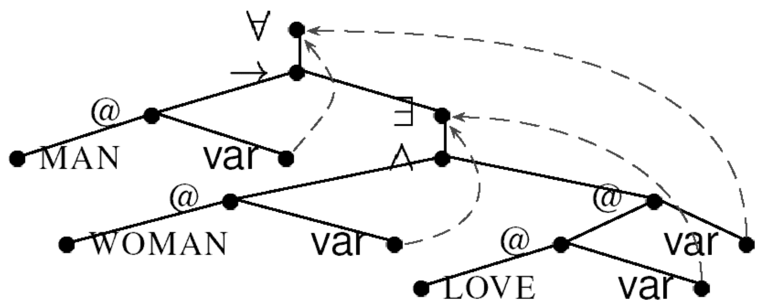
The picture so far



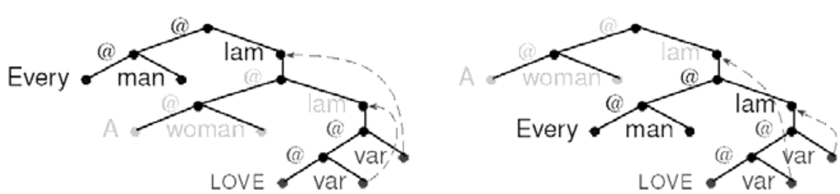
How to get readings?



Formulas and dominance structure

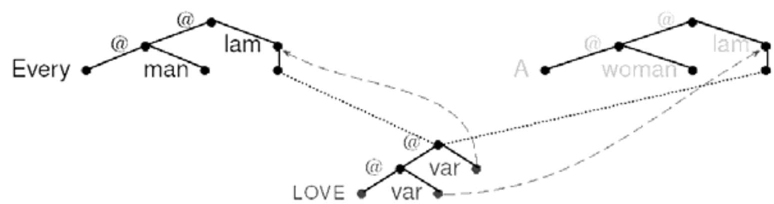


With lambda expressions



Take out shared material & convert to a <u>constraint graph</u>

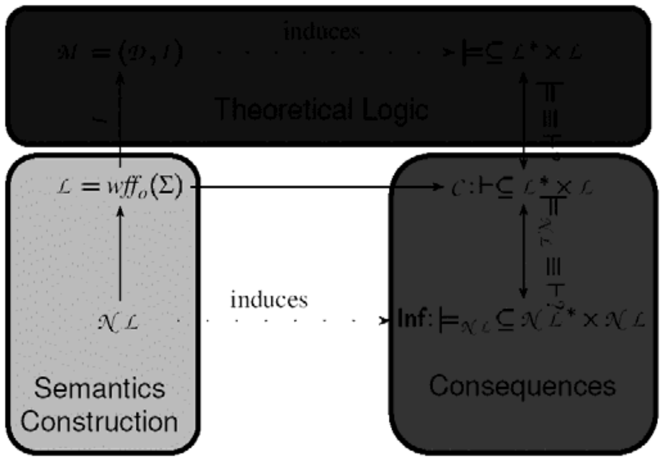
Constraint graph



Constraint graphs and l structures

- Any node that has a label in the graph must have the same label in the λ-structure.
- No two nodes that have a label in the graph must be mapped to the same node in the λ-structure.
- Any two nodes connected with a solid edge or a binding edge in the graph must be connected in the same way in the λ-structure.
- Whenever there is a dominance edge from a node X to a node Y in the graph, there must be a path from X to Y using only solid edges in the λ-structure.

The REALLY big picture – truth or consequences



But what about…

- Some linguist speaks at most 2 languages
  - Some linguist x is such that x speaks at most 2 languages
  - There are at most 2 languages y s.t. some linguist or other speaks those 2 languages y

And in general

- (Every x: Person(x)) (love(x, ice-cream))
- In general, the second component love(…) could be any predicate P (every person loves parfait…, every person hates dentists), where P is an arbitrary λ form
- So, we really need this:

  (Every x: Person(x)) P. x) and substitute P

Lambda abstract P:

λP (Every x: Person(x)) P. x)

Cognition as computation

- Computation manipulates formal symbols
- The symbols are represented
- The symbol manipulation is purely syntactic
- The symbol manipulation is semantically invariant

Our general view

- Syntactic representations to…
- Semantic representations to…
- Conceptual representations…

We know…

- What syntactic representations are
- We know much less about semantic or conceptual representations, but…
- Assume: they are the representations and vehicle for reasoning…
- So…must preserve what?
- Should be built up compositionally
- Why?

Compositionality, Turing, and all that

- Brown cow →
- Meaning(Brown) & Meaning(cow) & some mode of composition
- Why?
- Cf: Purple cow

Easy case

- Bob sleeps
- Bob likes ice-cream
- Event: likes(Bob, ice-cream)

Hard case

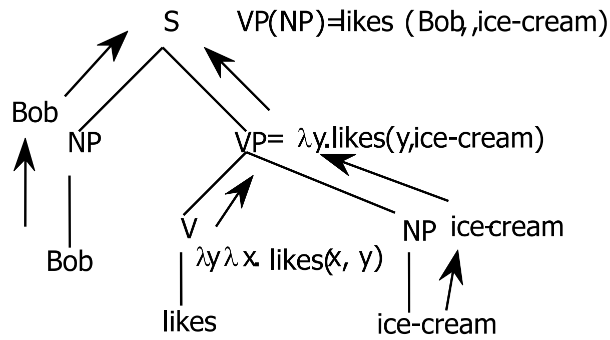
(But the Accord was redesigned for the 2003 model year.)

The roomier, faster, and sleeker sedan's sales stabilized last year,falling by just 1,230 units -- a strong showing in a market that saw combined total passenger car sales fall by 471,000 units.

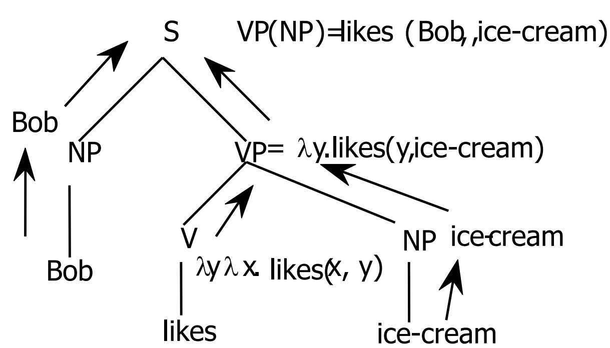
The envelope please…

the(x1,e1&e3&e5&e7) & more'(e1,x1,y1,e2) & roomy'(e2,x1)
& more'(e3,x1,y1,e4) & fast'(e4,x1) & more'(e5,x1,y1,e6) & sleek'(e6,x1)
& sedan'(e7,x1) & poss(x1,z1) & sale(z1,x2) & Plur(z1,s1)
& stabilize'(e8,s1) & Past(e8) & at-time(e8,y2) & last(y2,u1) & year(y2)
& fall'(e9,s1) & by(e9,s2) & just(e6) & card'(e6,s2,1230) & unit(u2) & Plur(u2,s2)
& Appos(e8,e11) & a(e11,e10&e11) & strong'(e10,e11) & show'(e11,x3,x4)
& in(e10,m) & a(m,e12&e13) & market'(e12,m) & see'(e13,m,e14) & Past(e13)
& combine(x5,s3) & total(s3) & passenger(p) & nn(p,c) & car(c)
& nn(c,z2) & sale(z2,x6) & Plur(z2,s3)
& fall'(e14,s3) & by(e14,s4) & card(s4,471000) & unit(u3) & Plur(u3,s4)

Why: recover meaning from structure – syntax-directed translation



How: function application



What's meaning? What's semantics – 2 ends of the spectrum

- Answer 1: whatever it is, it's mapping (translation) between representations
  And it depends on *all* of the text
- Answer 2: whatever it is, our answer depends on a much more focused task-specific question, viz., information extraction from texts
- Perhaps call this 'natural language engineering'

- These two ends of the spectrum have different characteristics, and difft uses
- Deep vs. Shallow?

What Counts as Understanding? some notions

- We understand statement if we know *how* to determine its truth
  - What are exact conditions under which it would be true?
    - necessary + sufficient
  - Equivalently, derive all its consequences
    - what else must be true if we accept the statement?
  - Philosophers tend to use this definition
- We understand statement if we can use it to answer questions [very similar to above – requires reasoning]
  - Easy: John ate pizza. What was eaten by John?
  - Hard: White's first move is P-Q4. Can Black checkmate?
  - Constructing a *procedure* to get the answer is enough

What Counts as Understanding?

- Be able to translate
  - Depends on target language
  - English to English? bah humbug!
  - English to French? reasonable
  - English to Chinese? requires deeper understanding
  - English to *logic*? deepest

  all humans are mortal = $\forall x\ [\mathrm{human}(x) \Rightarrow \mathrm{mortal}(x)]$

- Assume we have logic-manipulating rules to tell us how to act, draw conclusions, answer questions …

Answer 1: translation – from 'syntactic' rep to 'semantic' rep, aka "Deep"

- Mirrors the progamming language approach
- When is it used?
- DB Q&A (but answer 2 can be used here…when and how?)
- Text understanding: when *all* the text is relevant - voice, inference, paraphrase, important
- Intentions, beliefs, desires (non-extensional= not just sets of items)

What requirements must meaning representations fulfill?

- Verifiability: The system should allow us to compare representations to facts in a Knowledge Base (KB)
  - Cat(Huey)
- Ambiguity: The system should allow us to represent meanings unambiguously
  - 'German teachers' has 2 representations
- Vagueness: The system should allow us to represent vagueness
  - He lives somewhere in the south of France.

Requirements: Canonical Form

- Inputs that mean the same thing have the same representation.
  - Huey eats kibble.
  - Kibble, Huey will eat.
  - What Huey eats is kibble.
  - It's kibble that Huey eats.
- Alternatives
  - Four different semantic representations
  - Store all possible meaning representations in Knowledge Base

Requirements: Semantic Ambiguity

- Parallel to syntactic ambiguity
  - *Happy [cats and dogs] live on the farm*
  - *[Happy cats] and dogs live on the farm*
- Independent of syntactic structure
  - *Every boy loves a dog*
  - “all boys love a single dog”
  - “foreach boy, there is a dog he loves”

Requirements: Inference

- Draw valid conclusions based on the meaning representation of inputs and its store of background knowledge.

  Does Huey eat kibble?

  thing(kibble)

  Eat(Huey,x) ^ thing(x)

Word Senses & Ambiguity

- Q: Can the basic unit of meaning rep be a word?
- A: No, words have different *senses*
- Example: *go* has many senses (to move, depart, pass, vanish, reach, extend, …)
- Senses are organized into an *ontology*

Requirements: Word Senses

- Ontology
  - Example: Aristotle's classes
    - substance (physical objects)
    - quantity (e.g., numbers)
    - quality (e.g., being red)
    - Others: relation, place, time, position, state, action, affection
  - Important: actions, events
    - Provide a structure for organizing the interpretation of sentences

Requirements: Actions and Events

- *We lifted the box. It was hard work.*
  - The pronoun *it* refers to the whole action (not just *the box*)
- *We lifted the box. It was heavy.*
  - The pronoun *it* refers to *the box*

Need some kind of logical calculus

- Not ideal as a meaning representation and doesn't do everything we want - but close
  - Supports the determination of truth
  - Supports compositionality of meaning
  - Supports question-answering (via variables)
  - Supports inference
- What are its elements?
- What else do we need?

Logical Form Language

- Similar to first-order predicate calculus (FOPC)
- Constants: word senses
- Terms: constants that describe objects in the world
- Predicates: constants that describe relations or properties
- Propositions: predicate + terms

First order predicate calculus (FOPC)

Propositional logic: Don't look inside propositions: P, Q, R, ...
First-order logic: Look inside propositions: p(x,y), like(J,M), ...

Constants: John1, Sam1, ..., Chair-46, ..., 0, 1, 2, ...
Variables: x, y, z, ...
Predicate symbols: p, q, r, ..., like, hate, ...
Function symbols: motherOf, sumOf, ...
All the logical connectives of propositional logic.

Predicates and functions apply to a fixed number of arguments:
  Predicates: like(John1,Mary1), hate(Mary1,George1), tall(Sue3), ...
  Functions: motherOf(Sam1) = Mary1, sumOf(2,3) = 5, ...

In the expression: 3 + 2 > 4
function predicate

Predicates applied to arguments are propositions and yield True or False.
Functions applied to arguments yield entities in the domain.

Predicates

- *Fido is a dog*
  (DOG1 FIDO1)
  unary predicate
- *Sue loves Jack*
  (LOVES SUE1 JACK1) or LOVES(Sue, Jack)
  binary predicate
- We shall place this into an <u>event structure</u>:

Event(Loves1 :Agent Sue1 :Patient Jack1
Time: present)

Extension of a predicate

The semantics of a unary predicate is the set of all entities in the domain for which the predicate is true.

The predicate dog → the set of all dogs (in the real world)

This is the <u>extension</u> of the predicate dog.

That leaves out possible dogs, future dogs, etc.; makes dog-ness depend on 'accidental' historically contingent properties of the world

Possible Worlds

Possible world: A technical device in logic for handling "possible"

And a very powerful tool for analyzing some concepts.

You can use them without believing in them.

Duality between possible worlds and propositions:

A proposition can be viewed as the set of all possible worlds in which the proposition is true.

A possible world can be viewed as the set of all propositions that are true in it.

Add another proposition that has to be true
←→ Make the set of possible worlds smaller

Possible worlds to define 'intension' of a predicate

Intension: Map the predicate dog into a mapping from all possible worlds to the set of dogs in that possible world.

the predicate dog → [F: possible world w → the set of dogs in w

Given a predicate and a possible world, the intension will tell you the set of things that satisfy that predicate in that world.

Intension does a better job of capturing the essence of the concept.

A simple semantics for sentences

- Assuming that meaning of sentence is the proposition *p* expressed by sentence
- Simply its 'truth conditional' content, I.e,

  $p{:}w \rightarrow \{0,1\}$ (*w= 'a possible world'*)

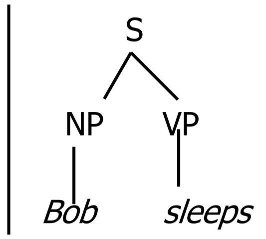
This function (the proposition *p* expressed by *s*) may be viewed as:

  - The truth conditions of a sentence s
  - Assigning the values 0 or 1 for any given *w*
  - Or as the set of possible worlds or situations where *s* is true

From syntactic structures to semantic structures

- We know what the structure of a simple subject-predicate sentence is
- We also know its meaning: the proposition of set of (all possible, not just actual) situations given by {*sit* | Peter sleeps in *sit*}
- Or: where individual denoted by "Peter" is in the extension of the predicate sleeps, I.e., in the set of all individuals that sleep

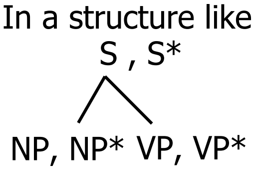
Syntax to semantics



→ SLEEP(Bob)

The master principles

- Compositionality
- In a structure like this:



- The meaning of the S is computed as the <u>function application</u> of the meaning of the VP to the meaning of the NP: S*=VP*(NP*)
- Intuitively: the concept expressed by the VP is asserted of the object to which the NP refers

The Principles

- Rule-to-rule hypothesis (Frege): semantic interpretation guided by syntactic structure;

For each syntactic rule, there is a corresponding rule of semantic interpretation

- Compositionality

We assume that the meaning of a complex expression is determined by the meaning of its parts

How to execute?

- Composition as function composition, I.e., function application
- We'll need a way to express this…
- Also need a way to express predicates generally…

NP meanings

- If just a common noun (CN), e.g., "Bob", "ice-cream", then it's like a constant

(i.e., picks out all the "Bobs" in the world…)

- We'll see how to express this in a moment…

VP meanings

- VP - sleeps (as intransitive)
- The meaning of the VP sleeps, then, is a function f from an individual x into a proposition (or a set of situations)

  $f(x) = \{situation \mid x \text{ sleeps in } situation\}$

How can we express this function?

6.001 to the rescue

- The function *f* can be given by the λ-expression

  λx SLEEPS(x)

- When this function is applied to the argument 'Bob', as usual this binds the variable x:

  λx SLEEPS(x)Bob → SLEEPS(Bob)

λ Abstraction to the rescue

- SLEEPS(BOB) is composed of the VP <u>meaning</u> which is the function λx SLEEPS(x), <u>applied to</u> an argument, the NP <u>meaning</u>, which is Bob

- Execution: associate with <u>each</u> context-free rule a corresponding <u>semantic rule</u>

Context-free semantics

Item or rule	Semantic translation
S → NP VP	S*: apply VP*(NP*)
VP → *sleeps*	VP*: λx SLEEPS(x)
NP → CN	NP*: λx.x
CN → *Bob*	CN*: 'Bob' (ie, a constant)

It all works…

S*: apply VP*(NP*)

λx SLEEPS(x) λx.x *'Bob'*

λx SLEEPS(x).Bob

SLEEPS(BOB)

OK, the next step… meaning of a transitive verb

- Bob likes ice-cream
- We already know the meaning of a VP likes *sleeps*, so we know the meaning of, e.g., '*likes ice-cream'*
- But what is the meaning of <u>likes</u>?
- {*situation* | Bob likes ice-cream in *situation* }
- We need a function that combines w/ ice-cream Goal: yield an intransitive VP meaning, as above,
- Intransitive: λx Likes-ice-cream(x)

Transitive verb meaning

- Intransitive: λx Likes-ice-cream(x)
- λy g(y) → LIKES(ice-cream)
- Lambda abstract:

  λy LIKES(y) for the VP
- Replace this in Likes-ice-cream(x):

  λx (λy LIKES(x, y)) or to fix order

  λy λx LIKES(x, y). ice-cream . Bob

This is the meaning of <u>likes</u>

Quantifier Ambiguities

Every person loves some desert.
==> (A p)(E i) love(p,d) i.e., chocolate cake
==> (E d)(A p) love(p,d) i.e., parfait

Most politicians in most countries can fool most of the people on most issues most of the time.

This has 120 possible readings, all distinct.
e.g., different issues for each country, or same issues?
different people for each issue, or same people?
Do we need to generate each separate reading?

The solution next time…!

- But there is a lot more to do…

What can we represent with this machinery?

- Is it enough?
- Is it too much?
- We have to look at natural language!

- Here are some 'classic' semantics and NL core issues

Representing Arguments

Jenny pushed the chair from the living room to the dining room for Sam yesterday

Arg. Agent Theme Source Goal Benefactor Time

Could represent this like
push(Jenny, Chair1, LR, DR, Sam, 21Jan04, ...)

Or like
push'(e) & Agent(Jenny,e) & Theme(Chair1,e) & Source(LR,e) & Goal(DR,e) & Benefactor(Sam,e) & atTime(e, 21Jan04)

Or like
push'(e, Jenny, Chair1) & from(e, LR) & to(e, DR) & for(e, Sam) & yesterday(e, ...)

from complements

from adjuncts

Equivalence of these: (A e,x,y)[push'(e,x,y) --> Agent(x,e) & Theme(y,e)]

Space, Time, Tense, and Manner

tense

John ran.
run'(e,J) & Past(e)

John ran on Tuesday.
run'(e,J) & Past(e) & onDay(e,d) & Tuesday(d)

John ran in Chicago.
run'(e,J) & Past(e) & in(e,Chicago)

John ran slowly.
run'(e,J) & slow(e)

John ran reluctantly.
run'(e,J) & reluctant(J,e)

Attributives

Some attributive adjectives have an implicit comparison set or scale:

A small elephant is bigger than a big mosquito.

That mosquito is big.
mosquito(x) & big(x, s)

The implicit comparison set or scale, which must be determined from context

Proper Names

Proper names:

Could treat them as constants:
Springfield is the capital of Illinois. → capital(Springfield, Illinois)

But there are many Springfields; we could treat it as a predicate true of any town named Springfield:
capital(x,y) & Springfield(x) & Illinois(y)

Or we could treat the name as a string, related to the entity by the predicate name:
capital(x,y) & name("Springfield", x) & name("Illinois", y)

Indexicals

An **indexical** or **deictic** is a word or phrase that requires knowledge of the situation of utterance for its interpretation.
"I", "you", "we", "here", "now", some uses of "this", "that", ...

The property of being "I" is being the speaker of the current utterance

Indexicals require an argument for the utterance or the speech situation.

I(x,u): x is the speaker of utterance u
you(x,u): x is the intended hearer of utterance u
we(s,u): s is a set of people containing the speaker of utterance u
here(x,u): x is the place of utterance u
now(t,u): t is the time of utterance u

from the quotation marks

Chris said, "I see you now."
==> say(Chris,u) & content(e,u) & see'(e,x,y) & I(x,u) & you(y,u) & atTime(e,t) & now(t,u)

Unreal Things

Herodotus worshipped Zeus.

(E e,h,z)
[worship'(e,h,z) & Past(e) & Herodotus(h) & Zeus(z)]

Herodotus and the worshipping existed in the past.
Did Zeus?

John wants to build a boat.

(E e1,j,e2,b)
[want'(e1,j,e2) & Present(e1) & John(j) & build'(e2,j,b) & boat(b)]

The wanting exists; the building doesn't (at least not yet).

Language talks about things that don't exist, and have various modalities of existence.

(E e,h,z) ... has to mean "There exists in some universe of possible individuals" ...
Existence in the real world has to be asserted separately, e.g., Present(e), Rexist(h)

Another Approach

Use "operators" that **scope** over existence.

(E j)[WANT(j, (E e,b)[build'(e,j,b) & boat(b)] & John(j)]

Special operator that takes logical expression as 2nd argument and blocks its evaluation to True or False

Exists only in John's wanting

This is a common approach to such modal concepts.
My view: unnecessarily complicates the logic, but mine is a minority view.

Intensions vs. Extensions

The semantics of a unary predicate
is the set of all entities in the domain for which the predicate is true

the predicate dog ==> the set of all dogs (in the real world)

This is the **extension** of the predicate dog.

That leaves out possible dogs, future dogs, etc.; makes dog-ness depend only on possibly accidental facts about the real world.

Intension: Map the predicate dog into a mapping from all possible worlds to the set of dogs in that possible world.

the predicate dog → [F: possible world w → the set of dogs in w

Given a predicate and a possible world, the intension will tell you the set of things that satisfy that predicate in that world.

Intension does a better job of capturing the essence of the concept.

Knowledge and Belief

Joan knows George is tall.
Joan believes Michael is tall.

Possible world treatment of knowledge:

True(p, w1): Proposition p is true in possible world w1
K(a, w1, w2): w2 is a world which is possible given what a knows in w1
(The w2's are the worlds in which a could be, given what a knows)

(A a,p,w1)[True(Know(a,p), w1) <--> (A w2)[K(a, w1, w2) --> True(p, w2)]]

A consequence of this is that you know all the logical consequences of what you know, e.g., all of the truths of mathematics if you know Peano's axioms.

Knowledge is true, justified belief.

Some Problems of Belief Contexts

My view: Possible worlds treatment of knowledge and belief complicate the logic too much.
Joan believes Michael is tall ==> believe(j,e) & tall'(e,m)

However there are problems (with this and every other treatment of belief):

De re vs. **de dicto** readings

Frege's **problem of identity**

Frege's Problem of Identy

Equals can be substituted for equals.

Joan believes the Evening Star is rising.

Evening Star = Morning Star (= Venus)

→ Joan believes the Morning Star is rising.

Possible solutions:

1. Block substitution of equals for equals in belief contexts:
   Then believe is an operator, not a predicate.
   Problems with keeping track of coreference:
   Joan believes the Evening Star is rising. It is bright.
2. They are not identical in all possible worlds; they just happen to be identical in the real world.
   Axiomatize "real-world-identical" in parallel with "=".
   A huge bother when dealing with coreference.
3. "The Evening Star" refers not to the real Evening Star, but to something like the concept of the Evening Star

Probably right

Frege's solution

Mutual Belief

A very important concept in dealing with language:
 We understand each other because we have a huge set of shared beliefs, and we build our utterances on this.
 We describe the new information we wish to convey in terms of the mutual beliefs we share with our hearers.

More than "common belief" (i.e., beliefs we both have);
 we have to believe we each believe the beliefs, and so on.

 I believe you believe I believe Bush believes he is president.

The properties of mutual belief:
 mb(s,p): The set of agents s believes proposition p

 mb(s,p) & member(a,s) --> believe(a,p)

 mb(s,p) --> mb(s, mb(s, p))

 These rules and the rules of logic are mutually believed.

Properties

Joan is taller than Mary → more(J, M, tall)
 i.e. treat predicates as individuals that can be arguments to other predicates

But what about

Joan is a better student than Mary, but Mary is a better tennis player.
 → more(J,M,good-student) & more(M,J,good-tennis-player)

These can get arbitrarily complex

Lambda abstraction:
 λ x [(E y)[good(x) & play(x,y) & tennis(y)]]

where [λ x p(x)](a) = p(a)

This allows us to represent arbitrarily complex predicates,
 but it takes us beyond first-order logic.

Other Uses of Properties

Opaque adjectives:

a toy car =/=> toy(x) & car(x)
a former president =/=> former(x) & president(x)
a fake diamond =/=> fake(x) & diamond(x)

Functional mapping a predicate into another predicate

More like: toy(car)(x); former(president)(x); fake(diamond)(x)

My notation: toy(e) & car'(e,x)
The adjective modifies the property of the head noun, not the referent of the head noun.

Verb phrase ellipsis:

John called his mother, and George did too.

λ x [call'(e,x,y) & mother(y,x)]

"the" and "a"

Conventional notation:

A car arrives. ==> (E x)[car(x) & arrive(x)]

The car arrives. ==> arrive(ι x [car(x)])

iota operator: **<u>the</u>** x such that car(x)

But "the" and "a" convey information:
"the": the entity referred to by the NP is mutually identifiable in context via the property conveyed by the rest of the NP.
The car is in the driveway. ← Known entity
"a": the entity referred to by the NP is *not* mutually identifiable in context via the property conveyed by the rest of the NP.
A car is in the driveway. ← New entity
John Kerry is a tall man. ← New property

My approach: the man → the(x,e) & man'(e,x)
a man → a(x,e) & man'(e,x)

Highly idiosyncratic

Sets and Plurals

Plurals require both a set and a representative or typical member of the set.

Joan saw numerous old friends at the conference.

describes the representative member of the set

describes the set

describes the property "friend"

Lots of technical problems

Past(e1) & see'(e, j, x) & numerous(s) & old(e) & friend'(e,x) & Plur(x,s)

Or:

(E s)[numerous(s) & (A x)[member(x,s) --> old(friend)(x,j) & see(j,x)]

the set

the representative member

Quantifier Ambiguities

Every person loves some desert.

==> (A p)(E i) love(p,d) i.e., mint fudge

==> (E d)(A p) love(p,d) i.e., parfait

Most politicians in most countries can fool most of the people on most issues most of the time.

This has 120 possible readings, all distinct.

e.g., different issues for each country, or same issues?

different people for each issue, or same people?

Do we need to generate each separate reading?

Monotone Decreasing Quantifiers

Quantifiers can be **monotone increasing**:

every man works hard ==> every man works
most men work hard ==> most men work
some men work hard ==> some men work

Make the predication more general, it's still true

Or **monotone decreasing**:

few men work hard =/=> few men work
no men work hard =/=> no men work

Make the predication more general, it's not necessarily true

Rather,

few men work ==> few men work hard
no men work ==> no men work hard

Make the predication more specific, it's still true

To make the flat notation work, we must interpret
Few men work
as
The men who work are few.

Donkey Sentences

Every man who owns a donkey beats it.

(A x)[(E y)[own(x,y) & donkey(y)] --> beat(x,y)]

But this reaches inside scope of existential quantifier

Concerns of sentences like this, and how to extend quantifiers beyond single sentences, have led to developments in semantics including
Discourse Representation Theory (DRT)
Dynamic logic

Negation

The car doesn't work.
~work(c) vs. not(e) & work'(e,j)

The car almost doesn't work.
ALMOST(~work(c)) vs. almost(e1) & not'(e1,e) & work'(e,c)

An operator,
not first-order

In fact, in not(e), e is really a representative element of a set

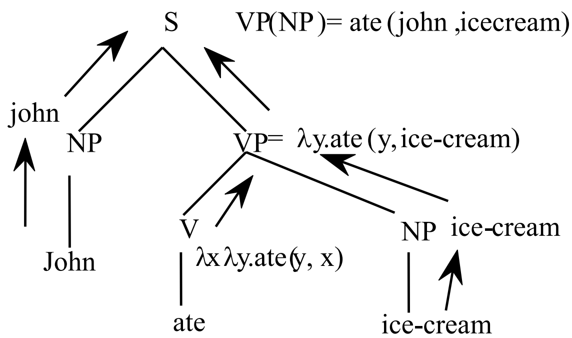
John didn't go to class. (yesterday)

To properly interpret negation, we have to figure out that set.

Whew…

- That's how hard it can get…
- Let's go back to the simple case.

Why: recover meaning from structure



"Logical" Form

- Context-independent meaning
  - Produced directly from the syntax
  - Ignores the utterance context
- Example: *The ball is red*
  - Assigning an exact (contextual) meaning requires knowing *which ball*
  - Logical form an *intermediate step* in full meaning representation

Logical Form [2]

- Includes *indexical terms*
  - Pronouns (e.g., *I, you*)
  - Generic NP (e.g., *a ball, the ball*)
  - Any term whose exact denotation can only be determined from context
- Logical form allows compact representation of indexical terms
  - e.g. (RED1 <THE b1 BALL>) vs. (OR b1 b4 b12 b45 …)

Word Senses & Ambiguity

- Q: Can the basic unit of LF be a word?
- A: No, words have different *senses*
- Example: *go* has many senses (to move, depart, pass, vanish, reach, extend, …)
- Senses are organized into an *ontology*

Logical Form Language

- Similar to first-order predicate calculus (FOPC)
- Constants: word senses
- Terms: constants that describe objects in the world
- Predicates: constants that describe relations or properties
- Propositions: predicate + terms

Predicates

- *Fido is a dog*
  (DOG1 FIDO1)
  unary predicate
- *Sue loves Jack*
  (LOVES1 SUE1 JACK1)
  binary predicate
- We shall place this into an event structure:

Event(Loves1 :Agent Sue1 :Patient Jack1
Time: present)

Word Senses

- Proper names: terms
  JACK1
- Common nouns: unary predicates
  (DOG1 <>)
- Verbs: n-ary predicates (really n?)
  (BREAK1 <> <>)

Operators

- Logical Operators
  - *not*, *or*, *and*, *if*, *only if*, …
- Logical form supports two kinds of operators:
  - as word senses (if the operator is part of the utterance)
  - as logical operators (if the operator isn't part of the utterance)

Operators [2]

- Examples
  - *Jack loves Sue or Jack loves Mary*
    (OR1 (LOVES1 JACK1 SUE1)(LOVES1 JACK1 MARY1))
  - *Jack loves Sue, Bill loves Mary*
    (& (LOVES1 JACK1 SUE1)(LOVES1 BILL1 MARY1))

Quantifiers

- FOPC: only universal and existential quantifiers: $\forall$, $\exists$
- English: much larger range: (Is this true?)
  - *all*, *some*, *most*, *many*, *a few*, *the*, …
- Generalized Quantifiers
  (<quantifier> <variable> : <restriction-proposition>
  <body-proposition>)

Quantifiers [2]

- *Most dogs bark*
  (MOST1 d1:(DOG1 d1)(BARKS1 d1))
- *Most barking things are dogs*
  (MOST1 d1:(BARKS d1)(DOG1 d1))
- *The dog barks*
  (THE x:(DOG1 x)(BARKS1 x))

Plural Forms [2]

- Distributive reading
  The dogs bark
  "There is a set of dogs, and each one barks"
- Collective reading
  The dogs met at the corner
  "*There is a set of dogs, and each one met at the corner"

Ambiguous Plurals

- Some sentences allow <u>both</u> collective and distributive readings

  Two guys bought a stereo
  "Each guy bought a stereo"
  "The two guys bought a stereo together"

This gets complex

- John ate an ice-cream in a booth
  - Event representation
  - ∃e past(e), act(e,eating), eater(e,John), exists(ice-cream, eatee(e)), exists(booth, location(e))
- John ate an ice-cream in every booth
  - ∃e past(e), act(e,eating), eater(e,John), exists(ice-cream, eatee(e)), <u>all</u>(booth, location(e)),

$\underbrace{\qquad\qquad\qquad}_{\exists g\ \text{ice-cream}(g),\ \text{eatee}(e,g)}$ $\underbrace{\qquad\qquad\qquad}_{\forall b\ \text{booth}(b) \Rightarrow \text{location}(e,b)}$

So this means..

- This means ∃e ∀b which means same event for every booth
- False unless John can be in every booth during his eating of a single ice-cream
- Which order do we want?
- ∃b ∀e: "for all booths b, there was such an event in b"

- Figuring this out requires a notion of <u>scope</u> (and so, structure…)
- But wait, there's more… what about *all, none, …*

Beliefs, Desires and Intentions

- How do we represent internal speaker states like believing, knowing, wanting, assuming, imagining..?
  - Not well modeled by a simple DB lookup approach
  - Truth in the world vs. truth in some possible world

  George imagined that he could dance.

  Geroge believed that he could dance.
- Augment FOPC with special modal operators that take logical formulae as arguments, e.g. believe, know

Intensions vs. Extensions

Last time talking about models, we said the semantics of a unary predicate is the set of all entities in the domain for which the predicate is true

the predicate dog ==> the set of all dogs (in the real world)

This is the **extension** of the predicate dog.

That leaves out possible dogs, future dogs, etc.; makes dog-ness depend only on possibly accidental facts about the real world.

Intension: Map the predicate dog into a mapping from all possible worlds to the set of dogs in that possible world.

the prediate dog ==> [F: possible world w ==> the set of dogs in w

Given a predicate and a possible world, the intension will tell you the set of things that satisfy that predicate in that world.

Intension does a better job of capturing the essence of the concept.

Intensional Arguments

- John wants a unicorn (cf., John wants an ice-cream)
    - "there is a unicorn u that Willy wants"
    - here the wantee is an individual entity
    - "Willy wants any entity u that satisfies the unicorn predicate"
    - here the wantee is a <u>type</u> of entity
- Problem
    - 'unicorn' is defined by the <u>set</u> of unicorns – its <u>extension</u>
    - BUT this set is empty
    - All empty sets are equal (but some are more equal than others…)
    - So, John wants a unicorn ≡ John wants a dodo
    - What's wanted (wantee) should be <u>intension</u> or criteria for being a unicorn
- (One) solution: possible world semantics:
    - Can imagine other worlds where set of unicorn ≠ set of dodos