A whirlwind introduction to computational semantics

Bill MacCartney

CS224U: Natural Language Understanding

7 February 2012

Reminder

Lit review due in one week!

Time to get cracking!

Where's the understanding?

- This course is supposed to be about natural language *understanding*, isn't it?
- But how much closer have we gotten to that goal?
  - With lexical semantic relations? WSD? Semantic role labeling? Relation extraction? Coreference resolution?
  - Arguably, all are necessary to NLU … but are they sufficient?
- Consider relation extraction
  - When successful, yields structured representation of meaning

    〈Bill Gates, founder, Microsoft〉
  - But what kinds of meanings cannot be expressed?

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?

A. Sculpture C is exhibited in room 1.
B. Sculpture H is exhibited in room 1.
C. Sculpture G is exhibited in room 2.
D. Sculptures C and H are exhibited in the same room.
E. Sculptures G and F are exhibited in the same room.

Travel reservations

Yes, hi, I need to book a flight for myself and my husband from Boston to SFO, or Oakland would be OK too. We need to fly out on Friday the 12th, and then I could fly back on Sunday evening or Monday morning, but he won't return until Wednesday the 18th, because he's staying for business. No flights with more than one stop, and we don't want to fly on United because we hate their guts.

Shallow vs. deep semantics

- We haven't tried to work out the meanings of complete sentences
- So we can't do everything we'd like
  - Not all tasks can ignore detailed sentence structure
  - Unsuitable if machine must *act*, rather than relying on user to interpret the author's meaning
- You get what you pay for:
  - Cheap, fast, low-level techniques are appropriate in domains where speed and volume matter more than accuracy
  - More computationally expensive, higher-level techniques are appropriate when higher-quality results are required

Early example: Chat-80

- Developed 1979-82 by Fernando Pereira & David Warren
- Proof-of-concept natural language interface to database
- Could answer questions about geography
- Implemented in Prolog
- Hand-built lexicon & grammar
- Highly influential NLIDB system

Things you could ask Chat-80

- Is there more than one country in each continent? No.
- What countries border Denmark? west_germany.
- What are the countries from which a river flows into the Black_Sea? [romania].
- What is the total area of countries south of the Equator and not in Australasia? 10228 ksqmiles.
- Which country bordering the Mediterranean borders a country that is bordered by a country whose population exceeds the population of India? turkey.
- How far is London from Paris? I don't understand!

The Chat-80 database

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

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

The Chat-80 grammar

```
/* Sentences */
sentence(S) --> declarative(S), terminator(.) .
sentence(S) --> wh_question(S), terminator(?) .
sentence(S) --> yn_question(S), terminator(?) .
sentence(S) --> imperative(S), terminator(!) .

/* Noun Phrase */
np(np(Agmt,Pronoun,[]),Agmt,NPCase,def,_,Set,Nil) -->
   {is_pp(Set)},
   pers_pron(Pronoun,Agmt,Case),
   {empty(Nil), role(Case,decl,NPCase)}.

/* Prepositional Phrase */
pp(pp(Prep,Arg),Case,Set,Mask) -->
   prep(Prep),
   {prep_case(NPCase)},
   np(Arg,_,NPCase,_,Case,Set,Mask).
```

Chat-80 demo

You can run Chat-80 yourself on the corn machines!

```
1. ssh corn
2. cd /afs/ir/class/cs224n/src/chat/
3. /usr/sweet/bin/sicstus
4. [load].
5. hi.
6. what is the capital of france?
```

Sample queries can be found at:

```
/afs/ir/class/cs224n/src/chat/demo
```

All the source code is there for your perusal as well

An NLU pipeline

- English sentences

  John smokes. Everyone who smokes snores.

- Syntactic analysis

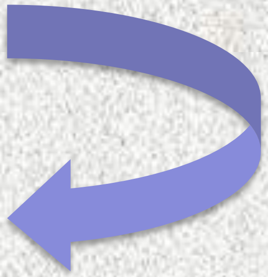
  (S (NP John) (VP smokes))

- Semantic analysis

  smoke(john)

- Inference

  $\forall x.\text{smoke}(x) \rightarrow \text{snore}(x), \text{smoke}(\text{john})$
  $\Rightarrow \text{snore}(\text{john})$



Focus of computational semantics

Meaning representations

- Many possible formal representations of meaning
  - DB tables, SQL, description logics, FOL, modal logics, …
- Blackburn & Bos (& others) argue for using FOL

John walks	walk(john)
John loves Mary	love(john, mary)
Every man loves Mary	$\forall x\ (\text{man}(x) \rightarrow \text{love}(x, \text{mary}))$

- But Manning focuses on lambda calculus — relationship?
  - We'll use lambda expressions for meanings of words & phrases
  - But for meanings of complete sentences, pure FOL — no lambdas

FOL syntax, in a nutshell

- FOL symbols
  - Constants: john, mary
  - Predicates & relations: man, walks, loves
  - Variables: *x*, *y*
  - Logical connectives: ∧ ∨ ¬ →
  - Quantifiers: ∀ ∃
  - Other punctuation: parens, commas

"content words" (user-defined)

"function words"

- FOL formulae
  - Atomic formulae: loves(john, mary)
  - Connective applications: man(john) ∧ loves(john, mary)
  - Quantified formulae: ∃*x* (man(*x*))

Compositional semantics

OK, we've chosen our target semantic representations

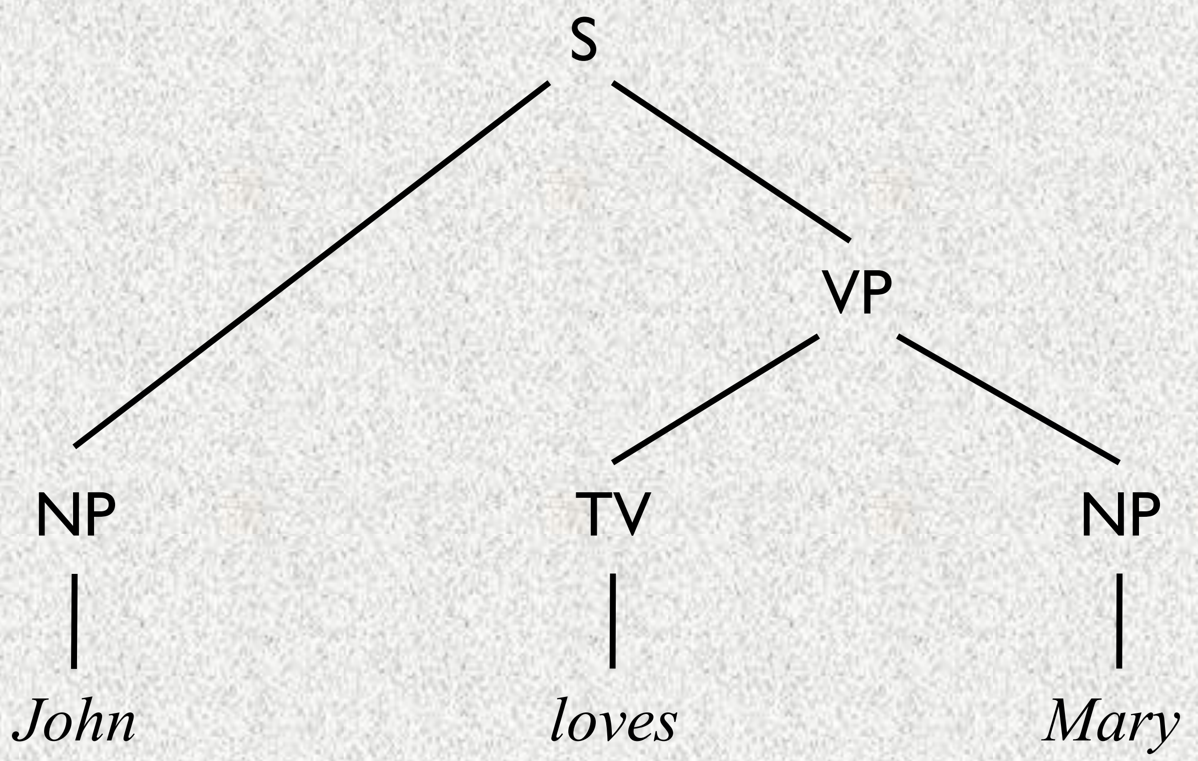
How do we derive them from raw input sentences?

1. Parse sentence to get syntax tree
2. Look up semantics of each word in lexicon
3. Build the semantics for each constituent
   - Bottom-up
   - Syntax-driven: "rule-to-rule translation"

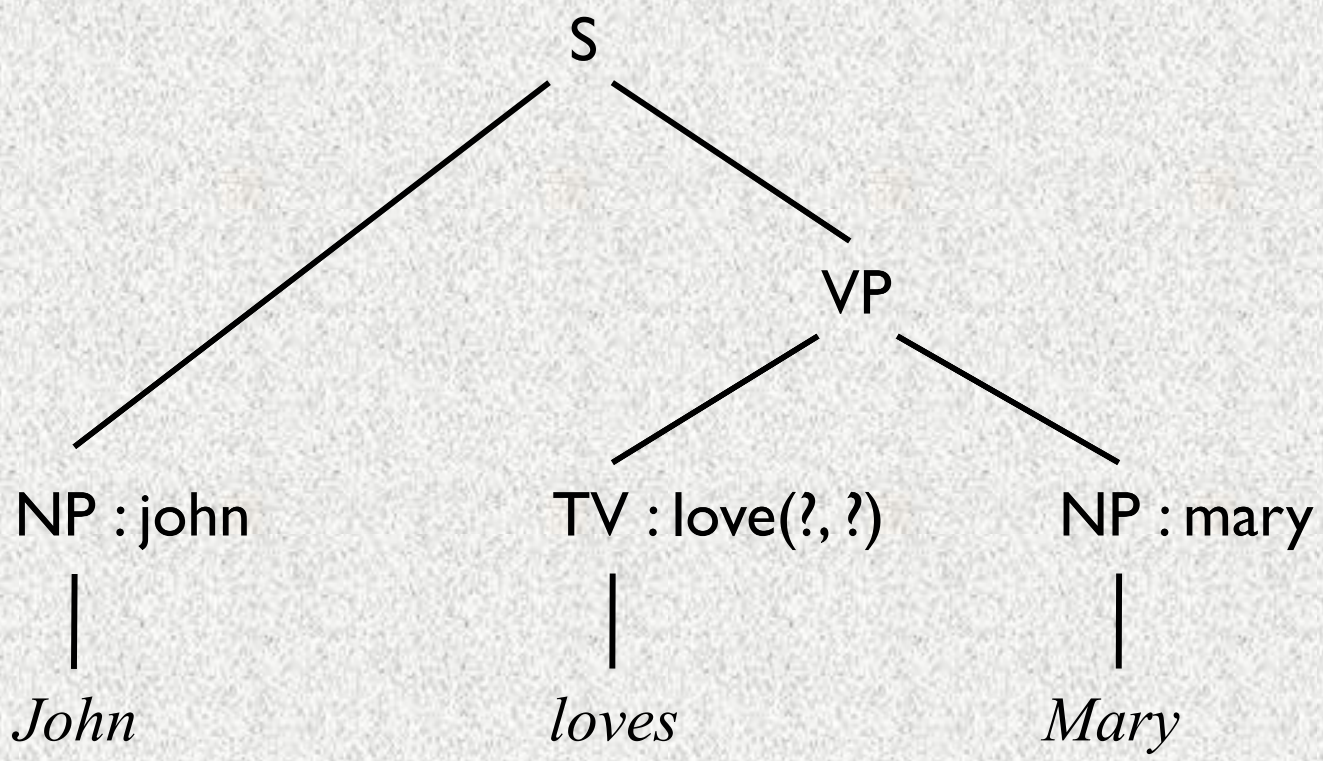
Principle of compositionality (aka Frege's Principle)

The meaning of a whole is determined by the meanings of the parts and the way in which they are combined.

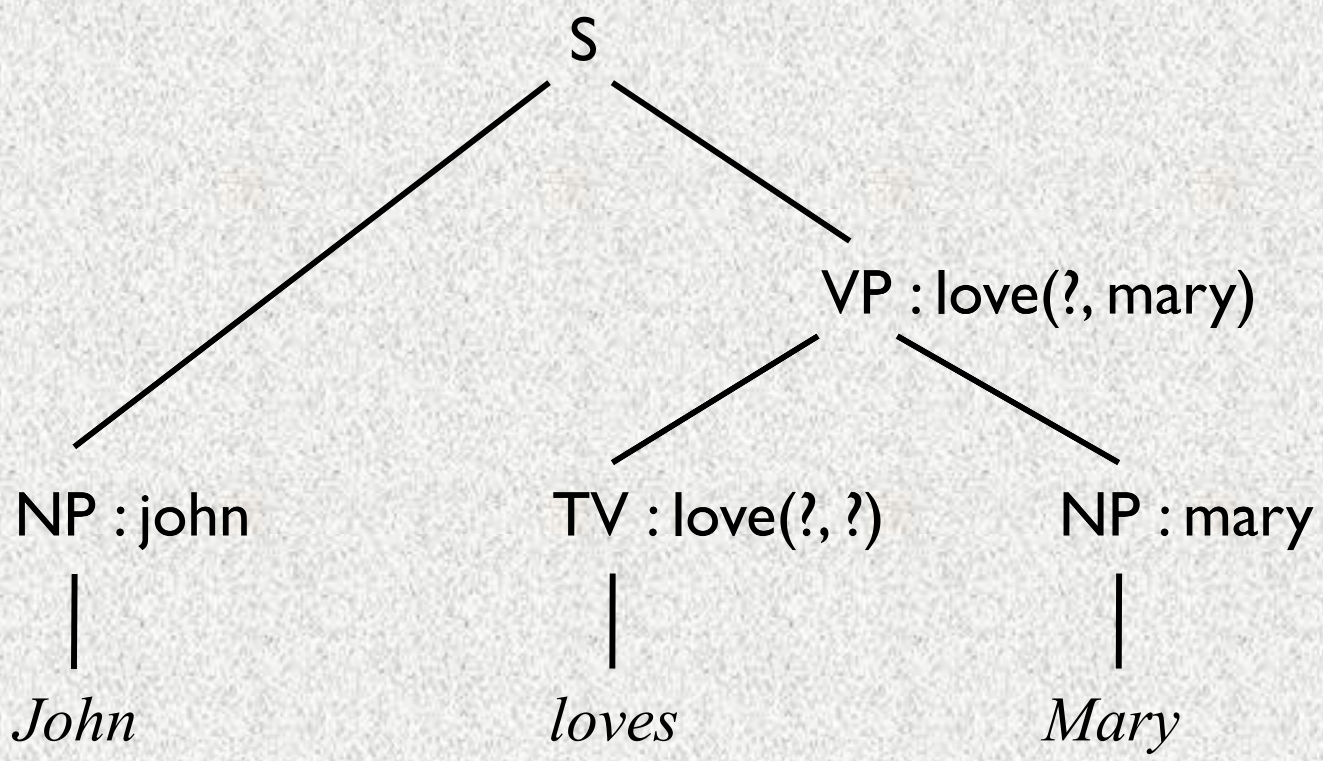
First example: syntactic analysis



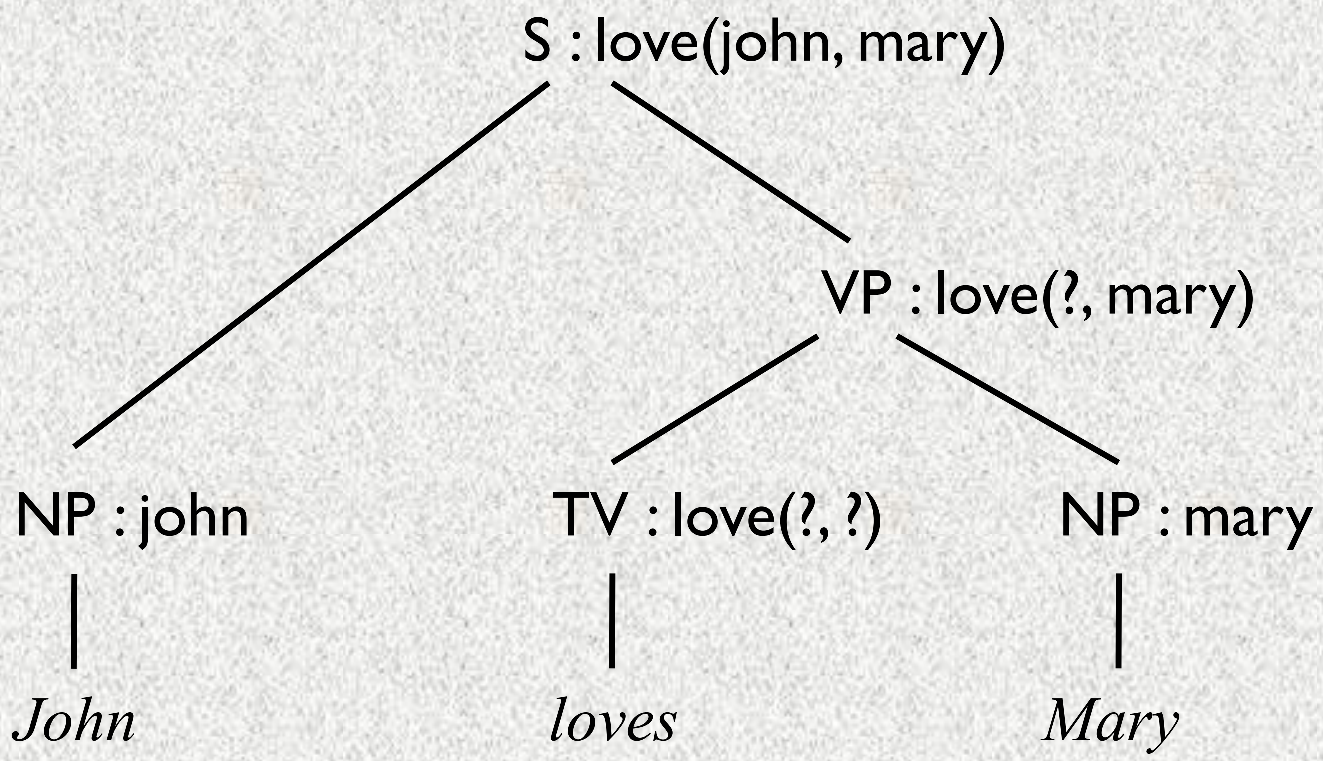
First example: semantic lexicon



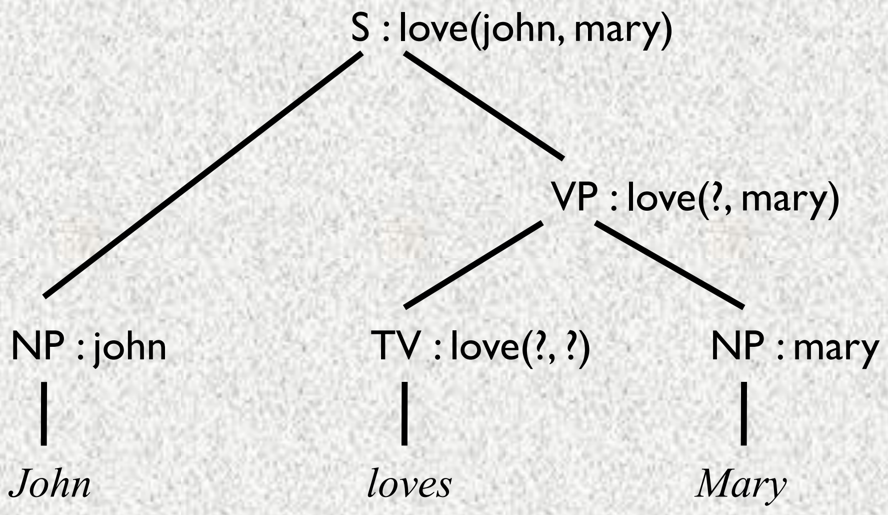
First example: semantic composition



First example: semantic composition



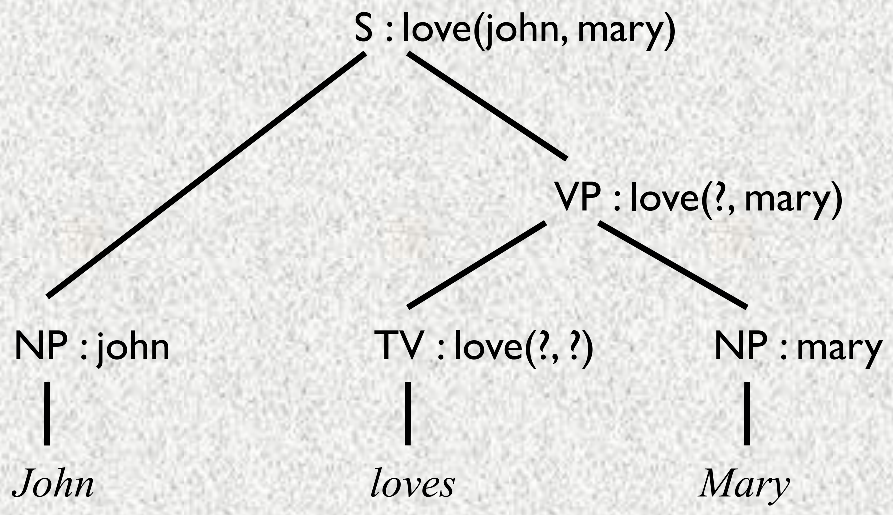
Compositionality



The meaning of the sentence is constructed from:

- the meaning of the words (i.e., the *lexicon*)
  john, mary, love(?, ?)
- paralleling the syntactic construction (i.e., the *semantic rules*)

Systematicity



How do we know how to construct the VP?

love(?, mary) OR love(mary, ?)

How can we *specify* in which way the bits & pieces combine?

Systematicity (continued)

- How do we want to represents parts of formulae?

  E.g. for the VP *loves Mary* ?

love(?, mary)	bad: not FOL
love(x, mary)	bad: no control over free variable

- Familiar well-formed formulae (sentences)

$\forall x\ (love(x, mary))$	*Everyone loves Mary*
$\exists x\ (love(mary, x))$	*Mary loves someone*

Lambda abstraction

- Add a new operator λ to bind free variables

  λx.love(x, mary)   *to love Mary*

- The new meta-logical symbol λ marks missing information in the object language (λ-)FOL
- We *abstract* over x
- Just like in programming languages!
  - Python: `lambda x: x % 2 == 0`
  - Ruby: `lambda {|x| x % 2 == 0}`
- How do we combine these new formulae and terms?

Super glue

- Gluing together formulae/terms with function application

  (λx.love(x, mary))   @   john

  (λx.love(x, mary))(john)

- How do we get back to the familiar love(john, mary) ?

- FA triggers a simple operation: *beta reduction*

  replace the λ-bound variable by the argument throughout the body

Beta reduction

(λx.love(x, mary)) (john)

1. Strip off the λ prefix

(love(x, mary)) (john)

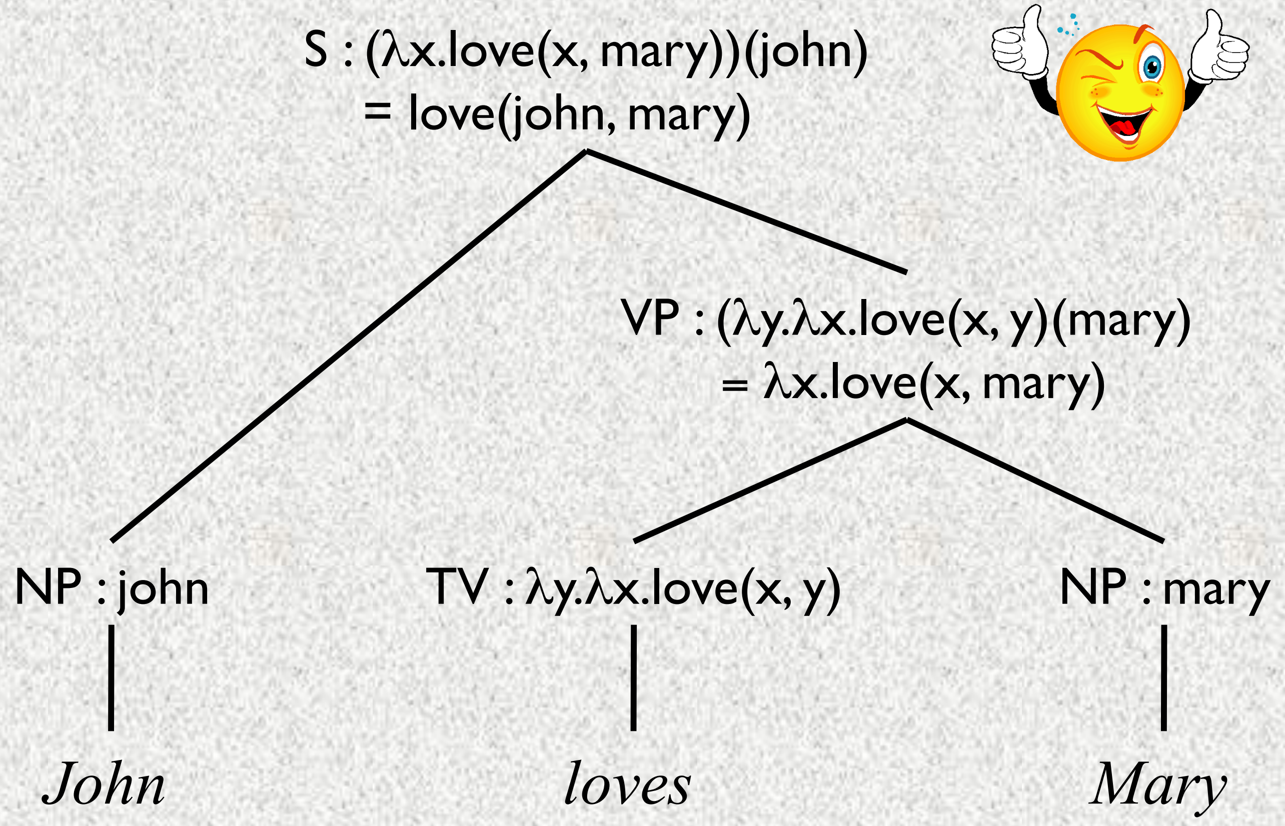
2. Remove the argument

love(x, mary)

3. Replace all occurrences of λ-bound variable by argument

love(john, mary)

Semantic construction with lambdas



A semantic grammar

Lexicon

John	$\leftarrow$	NP : john
Mary	$\leftarrow$	NP : mary
loves	$\leftarrow$	TV : $\lambda y.\lambda x.$love(x, y)

Composition rules

VP : f(a)	$\rightarrow$	TV : f	NP : a
S : f(a)	$\rightarrow$	NP : a	VP : f

Note the *semantic attachments* — these are augmented CFG rules

Note the use of function application to glue things together

For binary rules, four possibilities for semantics of parent (what?)

Montague semantics

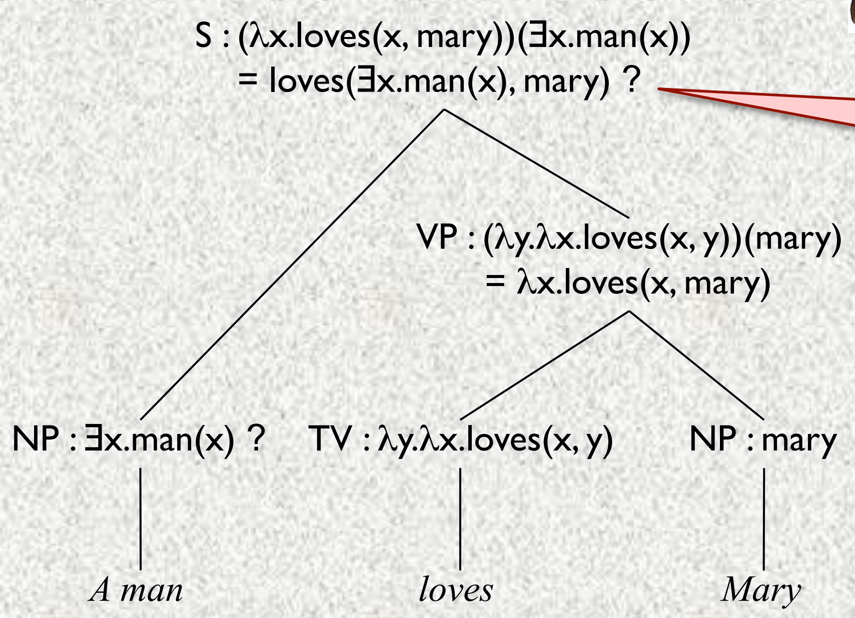
This approach to formal semantics was pioneered by Richard Montague (1930-1971)

"... I reject the contention that an important theoretical difference exists between formal and natural languages ..."

What about determiners?

How to handle determiners, as in *A man loves Mary*?

Maybe interpret "a man" as ∃x.man(x) ?


S : (λx.loves(x, mary))(∃x.man(x))
= loves(∃x.man(x), mary) ?
VP : (λy.λx.loves(x, y))(mary)
= λx.loves(x, mary)
NP : ∃x.man(x) ?
TV : λy.λx.loves(x, y)
NP : mary
A man
loves
Mary

How do we *know* this is wrong?

∃x.man(x) just doesn't mean "a man".

If anything it means "there is a man".

Analyzing determiners

Our goal is:

A man loves Mary $\rightarrow \exists z\ (\text{man}(z) \wedge \text{love}(z, \text{mary}))$

$\exists z\ ((\lambda y.\text{man}(y))(z) \wedge (\lambda x.\text{love}(x, \text{mary}))(z))$

What if we allow abstractions over any term?

$(\lambda Q.\exists z\ ((\lambda y.\text{man}(y))(z) \wedge Q(z)))\ (\lambda x.\text{love}(x, \text{mary}))$

$(\lambda P.\lambda Q.\exists z\ (P(z) \wedge Q(z)))\ (\lambda x.\text{love}(x, \text{mary}))\ (\lambda y.\text{man}(y))$

Add to lexicon:

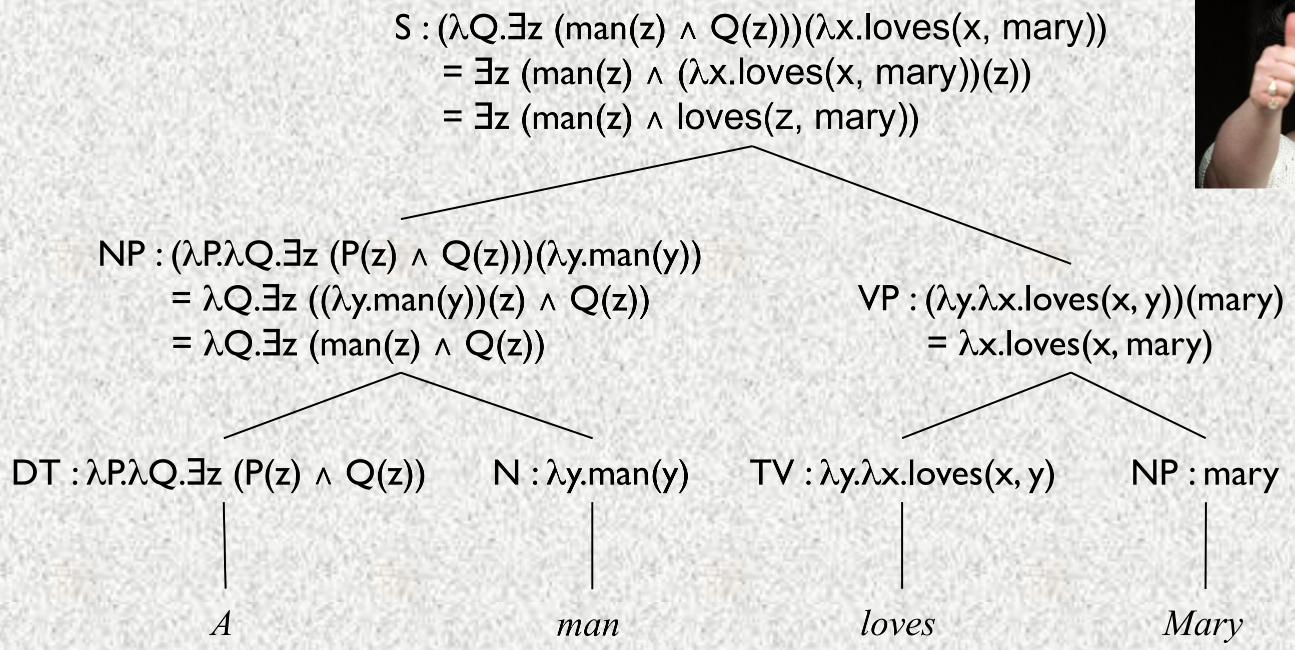
a $\rightarrow$ DT : $\lambda P.\lambda Q.\exists z\ (P(z) \wedge Q(z))$

And similarly:

every $\rightarrow$ DT : $\lambda P.\lambda Q.\forall z\ (P(z) \rightarrow Q(z))$

no $\rightarrow$ DT : $\lambda P.\lambda Q.\forall z\ (\neg P(z) \vee \neg Q(z))$

Determiners in action


$S : (\lambda Q.\exists z\ (man(z) \wedge Q(z)))(\lambda x.loves(x, mary))$
$= \exists z\ (man(z) \wedge (\lambda x.loves(x, mary))(z))$
$= \exists z\ (man(z) \wedge loves(z, mary))$
$NP : (\lambda P.\lambda Q.\exists z\ (P(z) \wedge Q(z)))(\lambda y.man(y))$
$= \lambda Q.\exists z\ ((\lambda y.man(y))(z) \wedge Q(z))$
$= \lambda Q.\exists z\ (man(z) \wedge Q(z))$
$VP : (\lambda y.\lambda x.loves(x, y))(mary)$
$= \lambda x.loves(x, mary)$
$DT : \lambda P.\lambda Q.\exists z\ (P(z) \wedge Q(z))$
$N : \lambda y.man(y)$
$TV : \lambda y.\lambda x.loves(x, y)$
$NP : mary$
A
man
loves
Mary

Add to lexicon

a	$\leftarrow$	$DT : \lambda P.\lambda Q.\exists z\ (P(z) \wedge Q(z))$
man	$\leftarrow$	$N : \lambda y.man(y)$

Add to grammar

$NP : f(a)$	$\leftarrow$	$DT : f \quad N : a$
$S : f(a)$	$\leftarrow$	$NP : f \quad VP : a$

different!

Hold up, wait a minute!

Now how are we going to handle *John loves Mary*?

(λx.love(x, mary)) @ (john)	not systematic!
(john) @ (λx.love(x, mary))	not reducible!
(λP.P(john)) @ (λx.love(x, mary))	better?
= (λx.love(x, mary))(john)	
= love(john, mary)	yes!

So revise lexicon:

John ← NP : λP.P(john)

Mary ← NP : λP.P(mary)

This is called *type-raising*:

old type: e   new type: (e→t)→t

The argument becomes the function!
(cf. callbacks, inversion of control)

Transitive verbs

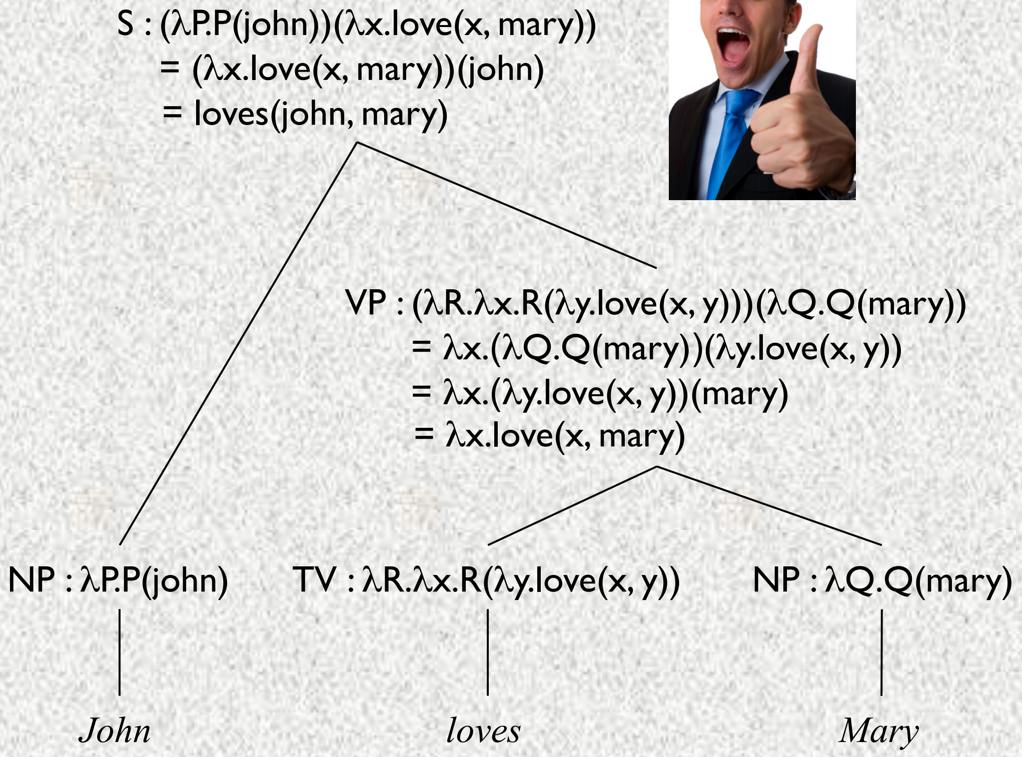
We had this in our lexicon:	*loves*	$\leftarrow$	TV : $\lambda y.\lambda x.\text{love}(x, y)$
But if we now have:	*Mary*	$\leftarrow$	NP : $\lambda P.P(\text{mary})$
then *loves Mary* will be	$(\lambda y.\lambda x.\text{love}(x, y))(\lambda P.P(\text{mary}))$		
	$= \lambda x.\text{love}(x, \lambda P.P(\text{mary}))$		

Uh-oh! Solution?

Type-raising again!	*loves*	$\leftarrow$	TV : $\lambda R.\lambda x.R(\lambda y.\text{love}(x, y))$
Old type for *loves*:	$e \rightarrow (e \rightarrow t)$		
New types for *loves*:	$((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$		

Let's see it in action …

Transitive verbs in action



S : $(\lambda P.P(john))(\lambda x.love(x, mary))$
$= (\lambda x.love(x, mary))(john)$
$= loves(john, mary)$

VP : $(\lambda R.\lambda x.R(\lambda y.love(x, y)))(\lambda Q.Q(mary))$
$= \lambda x.(\lambda Q.Q(mary))(\lambda y.love(x, y))$
$= \lambda x.(\lambda y.love(x, y))(mary)$
$= \lambda x.love(x, mary)$

NP : $\lambda P.P(john)$

TV : $\lambda R.\lambda x.R(\lambda y.love(x, y))$

NP : $\lambda Q.Q(mary)$

John

loves

Mary

Summing up

Our semantic lexicon covers many common syntactic types:

common nouns	*man*	←	$\lambda x.\text{man}(x)$
proper nouns	*Mary*	←	$\lambda P.P(\text{mary})$
intransitive verbs	*walks*	←	$\lambda x.\text{walk}(x)$
transitive verbs	*loves*	←	$\lambda R.\lambda x.R(\lambda y.\text{love}(x, y))$
determiners	*a*	←	$\lambda P.\lambda Q.\exists z(P(z) \wedge Q(z))$

We can handle multiple phenomena in a uniform way!

Key ideas:

- extra λs for NPs
- abstraction over (i.e., introducing variables for) predicates
- inversion of control: subject NP as function, predicate VP as arg

Coordination

How to handle coordination, as in *John and Mary walk*?

What we'd *like* to get:

$$\text{walk(john)} \wedge \text{walk(mary)}$$

Already in our lexicon:

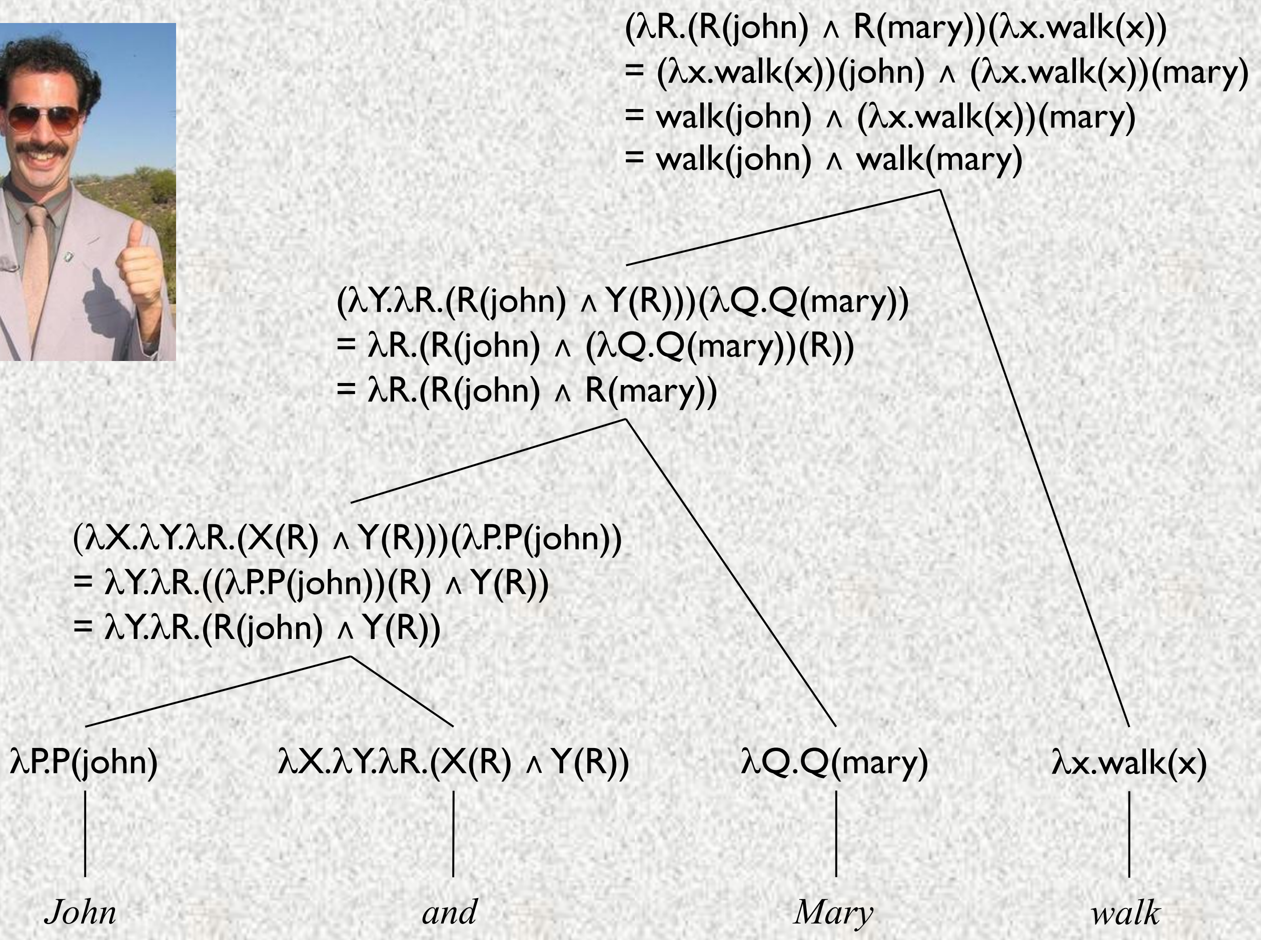
John	$\leftarrow$	NP : λP.P(john)
Mary	$\leftarrow$	NP : λQ.Q(mary)
walk	$\leftarrow$	IV : λx.walk(x)

Add to lexicon:

and	$\leftarrow$	CC : λX.λY.λR.(X(R) $\wedge$ Y(R))

My claim: this will work out just fine. Do you believe me?

Coordination in action



$(\lambda R.(R(john) \wedge R(mary))(\lambda x.walk(x))$
$= (\lambda x.walk(x))(john) \wedge (\lambda x.walk(x))(mary)$
$= walk(john) \wedge (\lambda x.walk(x))(mary)$
$= walk(john) \wedge walk(mary)$

$(\lambda Y.\lambda R.(R(john) \wedge Y(R)))(\lambda Q.Q(mary))$
$= \lambda R.(R(john) \wedge (\lambda Q.Q(mary))(R))$
$= \lambda R.(R(john) \wedge R(mary))$

$(\lambda X.\lambda Y.\lambda R.(X(R) \wedge Y(R)))(\lambda P.P(john))$
$= \lambda Y.\lambda R.((\lambda P.P(john))(R) \wedge Y(R))$
$= \lambda Y.\lambda R.(R(john) \wedge Y(R))$

$\lambda P.P(john)$ — *John*
$\lambda X.\lambda Y.\lambda R.(X(R) \wedge Y(R))$ — *and*
$\lambda Q.Q(mary)$ — *Mary*
$\lambda x.walk(x)$ — *walk*

Other kinds of coordination

So great! We can handle coordination of NPs!

But what about coordination of …

intransitive verbs	*drinks and smokes*
transitive verbs	*washed and folded the laundry*
prepositions	*before and after the game*
determiners	*more than ten and less than twenty*

One solution is to have multiple lexicon entries for *and*

We'll let you work out the details …

Quantifier scope ambiguity

In this country, a woman gives birth every 15 minutes.
Our job is to find that woman and stop her.

— Groucho Marx celebrates quantifier scope ambiguity

$\exists w\ (\text{woman}(w) \wedge \forall f\ (\text{fifteen-minutes}(f) \rightarrow \text{gives-birth-during}(w, f)))$

$\forall f\ (\text{fifteen-minutes}(f) \rightarrow \exists w\ (\text{woman}(w) \wedge \text{gives-birth-during}(w, f)))$

Surprisingly, both readings are available in English!

Which one is the joke meaning?

Where scope ambiguity matters!

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?

A. Sculpture C is exhibited in room 1.
B. Sculpture H is exhibited in room 1.
C. Sculpture G is exhibited in room 2.
D. Sculptures C and H are exhibited in the same room.
E. Sculptures G and F are exhibited in the same room.

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: At most three sculptures may be exhibited at all, regardless of which room.

Reading 3: The sculptures which can be exhibited in any room number at most three. (For the other sculptures, there are restrictions on 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

Classic example

Every man loves a woman

Reading 1: the women may be different

$\forall x\,(\mathrm{man}(x) \rightarrow \exists y\,(\mathrm{woman}(y) \wedge \mathrm{love}(x, y)))$

Reading 2: there is one particular woman

$\exists y\,(\mathrm{woman}(y) \wedge \forall x\,(\mathrm{man}(x) \rightarrow \mathrm{love}(x, y)))$

What does our system do?

Scope ambiguity in action

$$(\lambda Q.\forall z\,(\mathrm{man}(z) \rightarrow Q(z)))(\lambda x.\exists w\,(\mathrm{woman}(w) \wedge \mathrm{love}(x, w)))$$
$$= \forall z\,(\mathrm{man}(z) \rightarrow (\lambda x.\exists w\,(\mathrm{woman}(w) \wedge \mathrm{love}(x, w)))(z))$$
$$= \forall z\,(\mathrm{man}(z) \rightarrow \exists w\,(\mathrm{woman}(w) \wedge \mathrm{love}(z, w)))$$

$$(\lambda R.\lambda x.R(\lambda y.\mathrm{love}(x, y)))(\lambda Q.\exists w\,(\mathrm{woman}(w) \wedge Q(w)))$$
$$= \lambda x.(\lambda Q.\exists w\,(\mathrm{woman}(w) \wedge Q(w)))(\lambda y.\mathrm{love}(x, y))$$
$$= \lambda x.\exists w\,(\mathrm{woman}(w) \wedge (\lambda y.\mathrm{love}(x, y))(w))$$
$$= \lambda x.\exists w\,(\mathrm{woman}(w) \wedge \mathrm{love}(x, w))$$

$$(\lambda P.\lambda Q.\forall z\,(P(z) \rightarrow Q(z)))(\lambda y.\mathrm{man}(y))$$
$$= \lambda Q.\forall z\,((\lambda y.\mathrm{man}(y))(z) \rightarrow Q(z))$$
$$= \lambda Q.\forall z\,(\mathrm{man}(z) \rightarrow Q(z))$$

$$(\lambda P.\lambda Q.\exists w\,(P(w) \wedge Q(w)))(\lambda x.\mathrm{woman}(x))$$
$$= \lambda Q.\exists w\,((\lambda x.\mathrm{woman}(x))(w) \wedge Q(w))$$
$$= \lambda Q.\exists w\,(\mathrm{woman}(w) \wedge Q(w))$$

$\lambda P.\lambda Q.\forall z\,(P(z) \rightarrow Q(z))$ | $\lambda y.\mathrm{man}(y)$ | $\lambda R.\lambda x.R(\lambda y.\mathrm{love}(x, y))$ | $\lambda P.\lambda Q.\exists w\,(P(w) \wedge Q(w))$ | $\lambda x.\mathrm{woman}(x)$

Every *man* *loves* *some* *woman*

nltk.sem [Garrette & Klein 08]

The nltk.sem package contains Python code for:

- First-order logic & typed lambda calculus
- Theorem proving, model building, & model checking
- DRT & DRSs
- Cooper storage, hole semantics, glue semantics
- Linear logic
- A (partial) implementation of Chat-80!

http://nltk.googlecode.com/svn/trunk/doc/api/nltk.sem-module.html

nltk.sem.logic

```
>>> import nltk
>>> from nltk.sem import logic
>>> logic.demo()
>>> parser = logic.LogicParser(type_check=True)

>>> man = parser.parse("\ y.man(y)")
>>> woman = parser.parse("\ x.woman(x)")
>>> love = parser.parse("\ R x.R(\ y.love(x,y))")
>>> every = parser.parse("\ P Q.all x.(P(x) -> Q(x))")
>>> some = parser.parse("\ P Q.exists x.(P(x) & Q(x))")

>>> every(man).simplify()
<LambdaExpression \Q.all x.(man(x) -> Q(x))>

>>> love(some(woman)).simplify()
<LambdaExpression \x.exists z.(woman(z) & love(x, z))>

>>> every(man)(love(some(woman))).simplify()
<AllExpression all x.(man(x) -> exists z.(woman(z) & love(x, z)))>
```

What's missing?

OK, this all seems super duper, but … what's missing?

Can we solve these NLU challenges yet?

Why not?

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?

A. Sculpture C is exhibited in room 1.
B. Sculpture H is exhibited in room 1.
C. Sculpture G is exhibited in room 2.
D. Sculptures C and H are exhibited in the same room.
E. Sculptures G and F are exhibited in the same room.

Yes, hi, I need to book a flight for myself and my husband from Boston to SFO, or Oakland would be OK too. We need to fly out on Friday the 12th, and then I could fly back on Sunday evening or Monday morning, but he won't return until Wednesday the 18th, because he's staying for business. No flights with more than one stop, and we don't want to fly on United because we hate their guts.