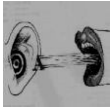


## Lecture 12: Semantics II



Professor Robert C. Berwick  
berwick@csail.mit.edu

## The Menu Bar

- Administrivia: Projects
- What knowledge do we need beyond syntax?
- How should we represent this?
- How can we learn it?
- From last time: representing ‘meaning’ as logical statements...we left off here...

6.863J/9.611J SP11

## Quantifier Order

- *Gilly swallowed a goldfish in a booth*
    - $\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{ swallowee}(e)), \text{ exists}(\text{booth}, \text{ location}(e)), \dots$
  - *Gilly swallowed a goldfish in every booth*
    - $\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{ swallowee}(e)), \text{ all}(\text{booth}, \text{ location}(e)), \dots$
- $\exists g \text{ goldfish}(g), \text{ swallowee}(e, g) \quad \forall b \text{ booth}(b) \Rightarrow \text{location}(e, b)$

- Does this mean what we'd expect??

says that there's only one event  
with a single goldfish getting swallowed  
that took place in a lot of booths ...

6.863J/9.611J SP11

## Quantifier Order

- Groucho Marx celebrates quantifier order ambiguity:
  - *In this country a woman gives birth every 15 minutes*
  - *Our job is to find that woman and stop her*
  - $\exists \text{woman} (\forall 15\text{min gives-birth-during}(\text{woman}, 15\text{min}))$
  - $\forall 15\text{min} (\exists \text{woman gives-birth-during}(15\text{min}, \text{woman}))$
  - Surprisingly, both are possible in natural language!
  - Which is the joke meaning (where it's always the same woman) and why?

6.863J/9.611J SP11

## Quantifier Order

- *Gilly swallowed a goldfish in a booth*
    - $\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{swallowee}(e)), \text{ exists}(\text{booth}, \text{location}(e)), \dots$
  - *Gilly swallowed a goldfish in every booth*
    - $\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{swallowee}(e)), \text{ all}(\text{booth}, \text{location}(e)), \dots$
- $\exists g \text{ goldfish}(g), \text{ swallowee}(e, g) \quad \forall b \text{ booth}(b) \Rightarrow \text{location}(e, b)$
- Does this mean what we'd expect??
    - It's  $\exists e \forall b$  which means same event for every booth
    - Probably false unless Gilly can be in every booth during her swallowing of a single goldfish

6.863J/9.611J SP11

## Intensional Arguments

- *Willy wants a unicorn*
  - $\exists e \text{ act}(e, \text{wanting}), \text{ wanter}(e, \text{Willy}), \text{ exists}(\text{unicorn}, \lambda u \text{ wantee}(e, u))$ 
    - “there is a unicorn  $u$  that Willy wants”
    - here the *wantee* is an individual entity
  - $\exists e \text{ act}(e, \text{wanting}), \text{ wanter}(e, \text{Willy}), \text{ wantee}(e, \lambda u \text{ unicorn}(u))$ 
    - “Willy wants any entity  $u$  that satisfies the unicorn predicate”
    - here the *wantee* is a type of entity
- *Willy wants Lilly to get married*
  - $\exists e \text{ present}(e), \text{ act}(e, \text{wanting}), \text{ wanter}(e, \text{Willy}), \text{ wantee}(e, \lambda e' [\text{act}(e', \text{marriage}), \text{ marrier}(e', \text{Lilly})])$
  - “Willy wants any event  $e'$  in which Lilly gets married”
  - Here the *wantee* is a type of event
  - Sentence doesn't claim that such an event exists
- Intensional verbs besides *want*: *hope, doubt, believe, ...*

6.863J/9.611J SP11

## Quantifier Order

- *Gilly swallowed a goldfish in a booth*
  - $\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{swallowee}(e)), \text{ exists}(\text{booth}, \text{location}(e)), \dots$
- *Gilly swallowed a goldfish in every booth*
  - $\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{swallowee}(e)), \text{ all}(\text{booth}, \lambda b \text{ location}(e, b))$
- Other reading ( $\forall b \exists e$ ) involves quantifier raising:
  - $\text{all}(\text{booth}, \lambda b [\exists e \text{ past}(e), \text{ act}(e, \text{swallowing}), \text{ swallower}(e, \text{Gilly}), \text{ exists}(\text{goldfish}, \text{swallowee}(e)), \text{ location}(e, b)])$
  - “for all booths  $b$ , there was such an event in  $b$ ”

6.863J/9.611J SP11

## Intensional Arguments

- *Willy wants a unicorn*
  - $\exists e \text{ act}(e, \text{wanting}), \text{ wanter}(e, \text{Willy}), \text{ wantee}(e, \lambda g \text{ unicorn}(g))$ 
    - “Willy wants anything that satisfies the unicorn predicate”
    - here the *wantee* is a type of entity
- Problem (a fine point I'll gloss over):
  - $\lambda g \text{ unicorn}(g)$  is defined by the actual set of unicorns (“extension”)
  - But this set is empty:  $\lambda g \text{ unicorn}(g) = \lambda g \text{ FALSE} = \lambda g \text{ dodo}(g)$
  - Then *wants a unicorn* = *wants a dodo*. Oops!
  - So really the *wantee* should be criteria for unicornness (“intension”)
- Traditional solution involves “possible-world semantics”
  - Can imagine other worlds where set of unicorns  $\neq$  set of dodos

6.863J/9.611J SP11

## Control

- *Willy wants Lilly to get married*
  - $\exists e$  present(e), act(e,wanting), wantee(e,Willy), wantee(e,  $\lambda f$  [act(f,marriage), marrier(f,Lilly)])
- *Willy wants to get married*
  - Same as *Willy wants Willy to get married*
  - Just as easy to represent as *Willy wants Lilly ...*
  - The only trick is to construct the representation from the syntax. The empty subject position of “to get married” is said to be controlled by the subject of “wants.”

6.863J/9.611J SP11

## Nouns and Their Modifiers

- *expert*
  - $\lambda g$  expert(g)
- *big fat expert* (cf, *big fat Greek wedding...*)
  - $\lambda g$  big(g), fat(g), expert(g)
  - But: *bogus expert*
    - Wrong:  $\lambda g$  bogus(g), expert(g)
    - Right:  $\lambda g$  (bogus(expert))(g) ... *bogus* maps to new concept
- *Boston expert (white-collar expert, TV expert ...)*
  - $\lambda g$  Related(Boston, g), expert(g) – expert from Boston
  - Or with different intonation:
    - $\lambda g$  (Modified-by(Boston, expert))(g) – expert on Boston
  - Can't use Related for that case: *law expert and dog catcher*
    - =  $\lambda g$  Related(law,g), expert(g), Related(dog, g), catcher(g)
    - = *dog expert and law catcher*

6.863J/9.611J SP11

## Nouns and Their Modifiers

- *the goldfish that Gilly swallowed*
  - *every goldfish that Gilly swallowed*
  - *three goldfish that Gilly swallowed*
- $\lambda g$  [goldfish(g), swallowed(Gilly, g)]
- *three <sup>like an adjective!</sup>swallowed-by-Gilly goldfish*
- Or for real:  $\lambda g$  [goldfish(g),  $\exists e$  [past(e), act(e,swallowing), swallower(e,Gilly), swallowee(e,g) ]]

6.863J/9.611J SP11

## Adverbs

- *Lili passionately wants Billy*
  - Wrong?: passionately(want(Lili,Billy)) = passionately(true)
  - Better: (passionately(want))(Lili,Billy)
  - Best:  $\exists e$  present(e), act(e,wanting), wantee(e,Lili), wantee(e, Billy), manner(e, passionate)
- *Lili often stalks Billy*
  - (often(stalk))(Lili,Billy)
  - many(day,  $\lambda d$   $\exists e$  present(e), act(e,stalking), stalker(e,Lili), stalkee(e, Billy), during(e,d))
- *Lili obviously likes Billy*
  - (obviously(like))(Lili,Billy) – one reading
  - obvious(likes(Lili, Billy)) – another reading

6.863J/9.611J SP11

## Speech Acts

- What is the meaning of a full sentence?
  - Depends on the punctuation mark at the end. ☺
  - Billy likes Lili. → **assert**(like(B,L))
  - Billy likes Lili? → **ask**(like(B,L))
    - or more formally, “Does Billy like Lili?”
  - Billy, like Lili! → **command**(like(B,L))
- Let’s try to do this a little more precisely, using event variables etc.

6.863J/9.611J SP11

## Standard answer to the last question, at least

- Syntax directed translation
- Output: instructions to do something in terms of some machine model
- Note the principles involved:
  - For each syntactic rule, there’s a corresponding local semantic translation rule
  - Compositionality: meaning of whole is *entirely* a function of the meaning of its parts

6.863J/9.611J SP11

## Speech Acts

- *What did Gilly swallow?*
  - **ask**( $\lambda x \exists e \text{ past}(e), \text{act}(e, \text{swallowing}), \text{swallower}(e, \text{Gilly}), \text{swallowee}(e, x)$ )
  - Argument is identical to the modifier “that Gilly swallowed”
  - Is there any common syntax?
- *Eat your fish!*
  - **command**( $\lambda f \text{ act}(f, \text{eating}), \text{eater}(f, \text{Hearer}), \text{eatee}(\dots)$ )
- *I ate my fish.*
  - **assert**( $\exists e \text{ past}(e), \text{act}(e, \text{eating}), \text{eater}(f, \text{Speaker}), \text{eatee}(\dots)$ )
- How do we map from syntax to this kind of semantics???

6.863J/9.611J SP11

## Hard case

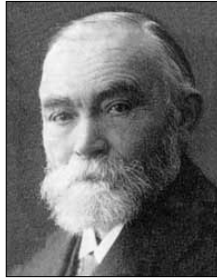
(But the Accord was redesigned for the 2003 model year.)

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.

6.863J/9.611J SP11

## 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

6.863J/9.611J SP11

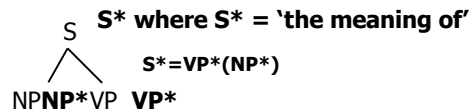
Context-free semantics: associate interpretation rule with each context-free rule

<u>Item or rule</u>	<u>Semantic translation</u>
$S \rightarrow NP VP$	<b>S*:</b> apply <b>VP*(NP*)</b>
$VP \rightarrow sleeps$	<b>VP*:</b> $\lambda x$ SLEEPS(x)
$NP \rightarrow Name$	<b>NP*:</b> $\lambda x$ x
$Name \rightarrow rocky$	<b>Name*:</b> 'rocky' (ie, a constant)

6.863J/9.611J SP11

## The master principles

- Compositionality
- In a structure like this:



- The meaning of the S is computed as the function application of the meaning of the VP to the meaning of the NP:
- Intuitively: the concept expressed by the VP is asserted of the object to which the NP refers

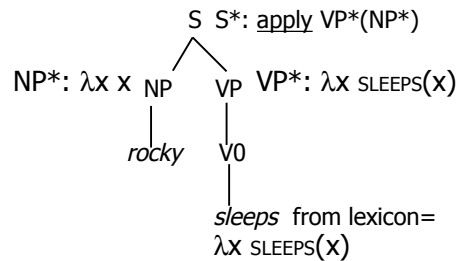
6.863J/9.611J SP11

## 6.001 to the rescue

- The function  $f$  can be given by the  $\lambda$ -expression  
 **$\lambda x$  SLEEPS(x)**
- When this lambda abstraction - a function - is applied to the argument 'Poirot', as usual this binds the variable  $x$ :  
 **$\lambda x$  SLEEPS(x).rocky  $\rightarrow$  SLEEPS(rocky)**
- The lambda variable acts as a placeholder for missing info
- The "**rocky**" denotes the substitution
- Alternatively, to make this more visible, we write:  
 **$\lambda x$  SLEEPS(x)**@**rocky  $\rightarrow$  SLEEPS(rocky)**
- Where the "**@**" denotes function application
- This process of substitution is also called beta reduction

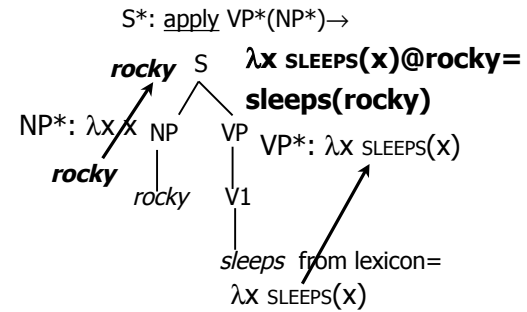
6.863J/9.611J SP11

Rule-to-rule principle: decorate syntax tree with



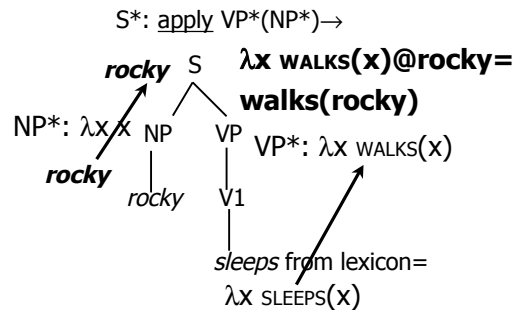
6.863J/9.611J SP11

Doing the beta reductions...



6.863J/9.611J SP11

What about 'rocky walks'?



Q1: in general, how can we abstract over all such intransitive verbs??? (answer in a bit)

6.863J/9.611J SP11

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 output will be some sort of first-order predicate calculus, or its 'equivalent' in SQL terms: e.g.,

"Rocky likes Bullwinkle" → LIKES (Rocky, Bullwinkle)

6.863J/9.611J SP11

## DB – the ‘truth’ as SQL

Rocky likes Bullwinkle →

**Insert into Like(likier, like) values(r, b)**

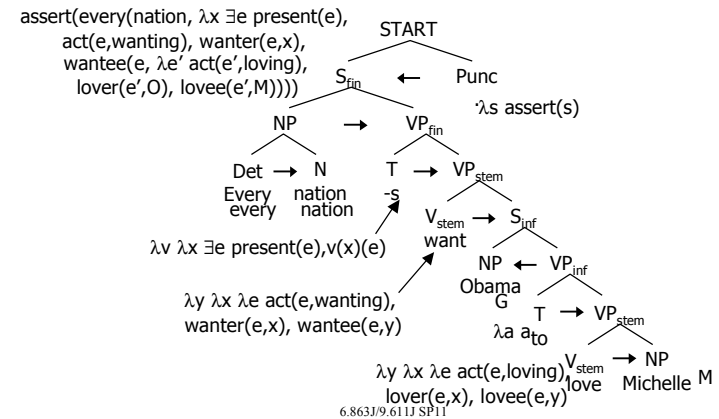
- Then we can answer a question, e.g.,

Does Rocky like Bullwinkle → via the SQL:

**select ‘yes’ from Like where Like.likier = r  
and Like.likee = b**

6.863J/9.611J SP11

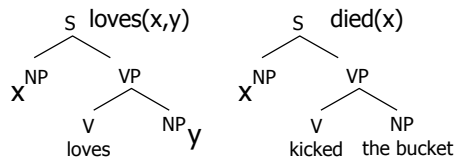
## Compositional Semantics



## Compositional Semantics

- Add a “sem” feature to each context-free rule
  - $S \rightarrow NP \text{ loves } NP$
  - $S[\text{sem}=\text{loves}(x,y)] \rightarrow NP[\text{sem}=x] \text{ loves } NP[\text{sem}=y]$
  - Meaning of S depends on meaning of NPs

• TAG version:



- Template filling:  $S[\text{sem}=\text{showflights}(x,y)] \rightarrow$   
I want a flight from  $NP[\text{sem}=x]$  to  $NP[\text{sem}=y]$

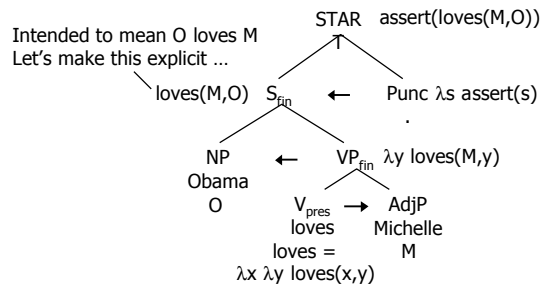
6.863J/9.611J SP11

## Compositional Semantics

- Instead of  $S \rightarrow NP \text{ loves } NP$ 
  - $S[\text{sem}=\text{loves}(x,y)] \rightarrow NP[\text{sem}=x] \text{ loves } NP[\text{sem}=y]$
- might want general rules like  $S \rightarrow NP \text{ VP}$ :
  - $V[\text{sem}=\text{loves}] \rightarrow \text{loves}$
  - $VP[\text{sem}=\text{v}(\text{obj})] \rightarrow V[\text{sem}=\text{v}] \text{ NP}[\text{sem}=\text{obj}]$
  - $S[\text{sem}=\text{vp}(\text{subj})] \rightarrow NP[\text{sem}=\text{subj}] \text{ VP}[\text{sem}=\text{vp}]$
- Now *Obama loves Michelle has*  
 $\text{sem}=\text{loves}(\text{Michelle})(\text{Obama})$ 
  - To get its semantics, apply function to argument!

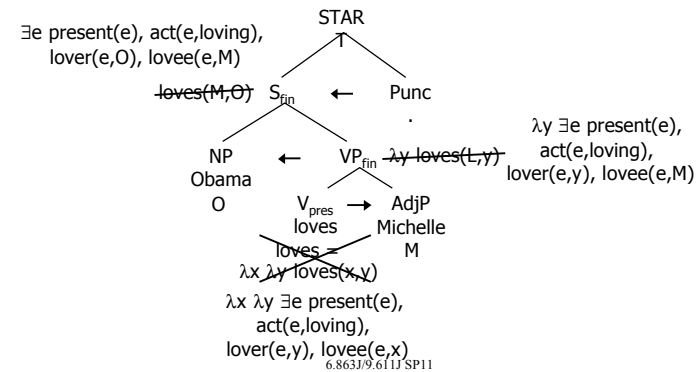
6.863J/9.611J SP11

## Compositional Semantics



6.863J/9.611J SP11

## Compositional Semantics



6.863J/9.611J SP11

## Names for things: one more trick needed

- So far we have names (constants) like 'rocky'
- Instead of "rocky" referring to an **Individual**, it will refer to the set of this individual's properties (viz., the predicates it satisfies, e.g., sleeps, walks, squirrel,...)
- Each property is a set, so this 'proper name' denotes a set of sets
- We will convert this to a function, in the following way:
- A proper name like Rocky will have the form:

$(\lambda P P)\text{@name}$ , where  $(\lambda P P)$  denotes some predicate to be supplied later

$\lambda P P\text{@rocky}$  = a *set* of sets

- This is called type raising because now, instead of 'rocky' being of a type "**Individual**" it is of a higher type (a function, that maps predicates P to truth values)

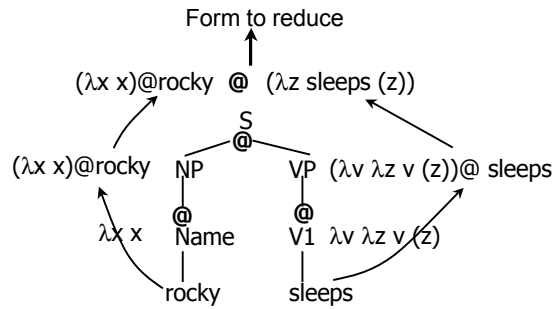
6.863J/9.611J SP11

## Names & intransitive verbs

- This trick lets us 'flip' functions and arguments: instead of  $(\mathbf{f} \mathbf{a})$  we have:  $(\lambda P (P \mathbf{a}) \mathbf{f})$
- We need this because sometimes the syntax of the natural language sentence doesn't have the structure (function argument) in the output semantic form, or vice-versa
- For intransitive verbs, in general, we can abstract out the predicate i.e. for an intransitive verb,  $v$ :  
 $\lambda v \lambda z v (z)$
- Now, every RHS rule with a single 'preterminal' adds a function application @ plus the preterminal, while every RHS with two nonterminals simply 'glues' the two pieces from below together, this way:

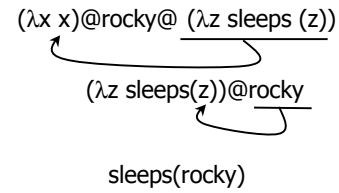
6.863J/9.611J SP11

We can now assemble full lambda form mechanically

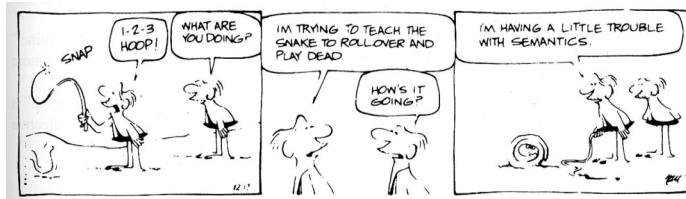


6.863J/9.611J SP11

Take this form and do beta-reductions (function applications) mechanically...

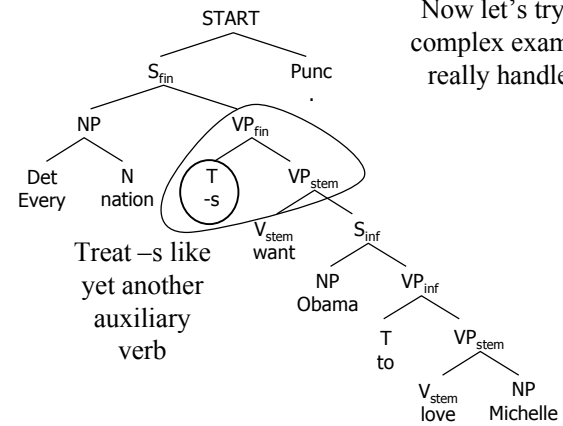


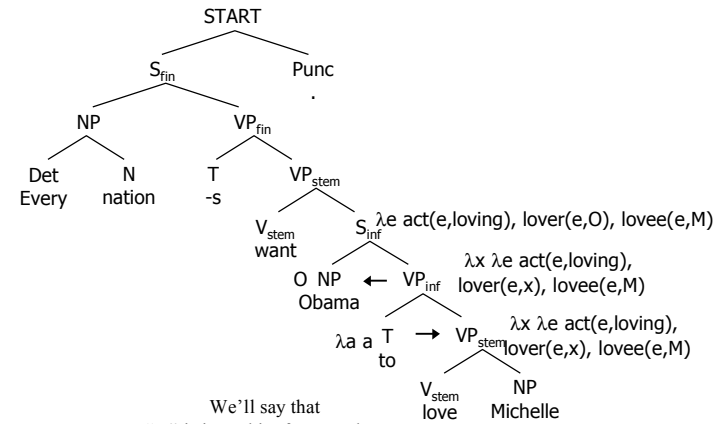
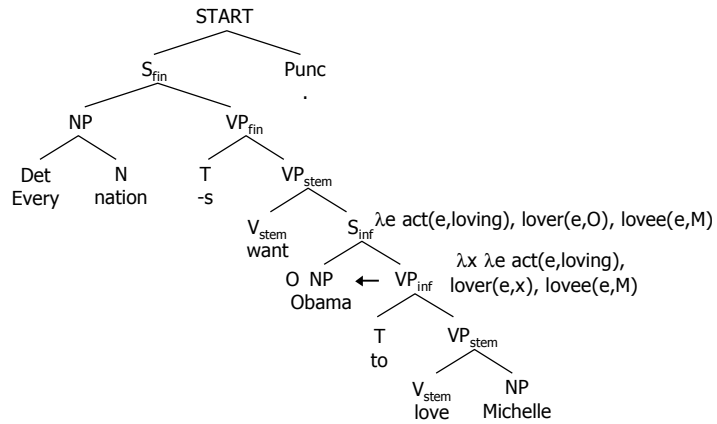
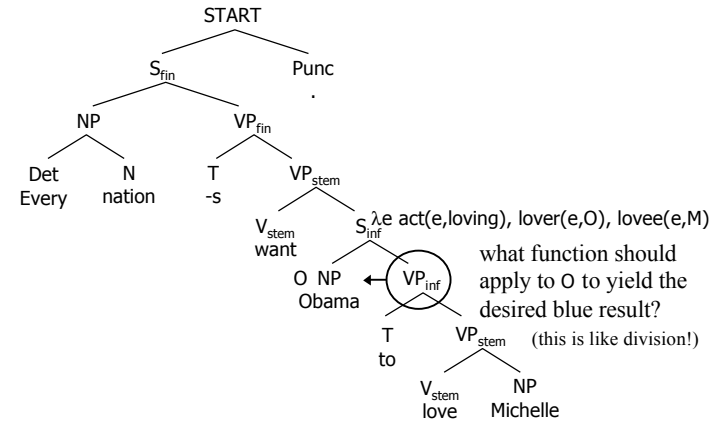
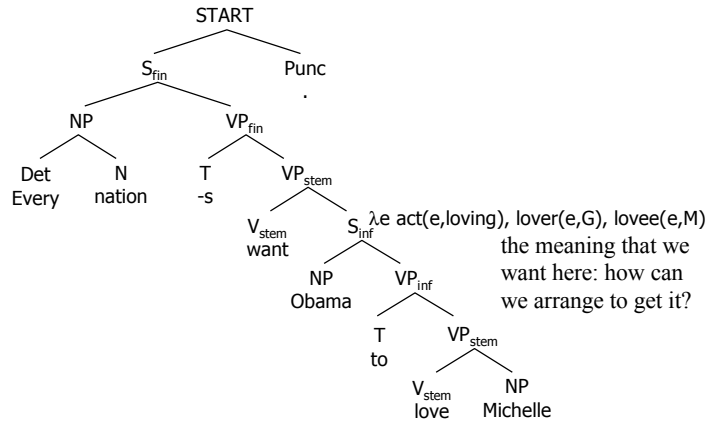
6.863J/9.611J SP11

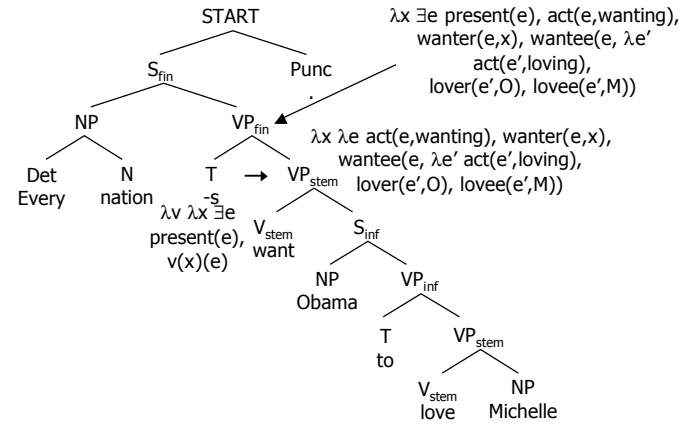
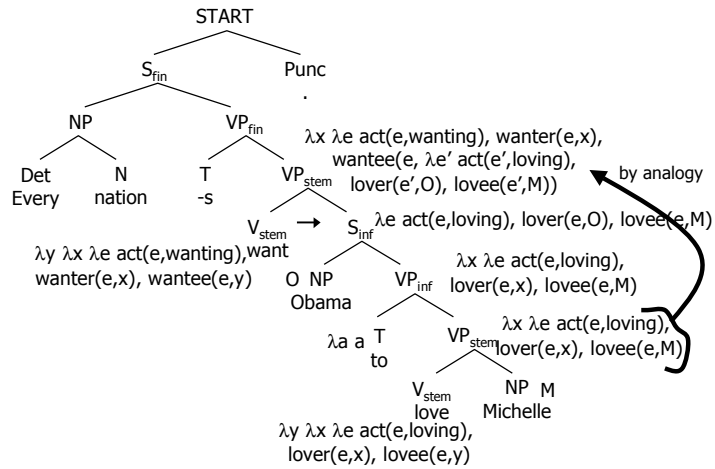
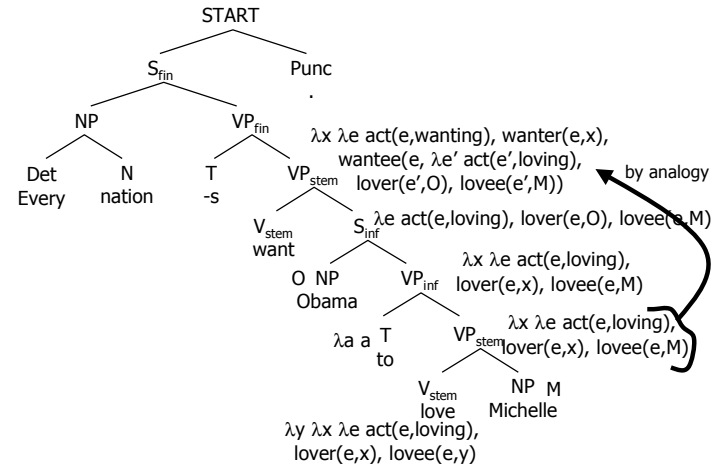
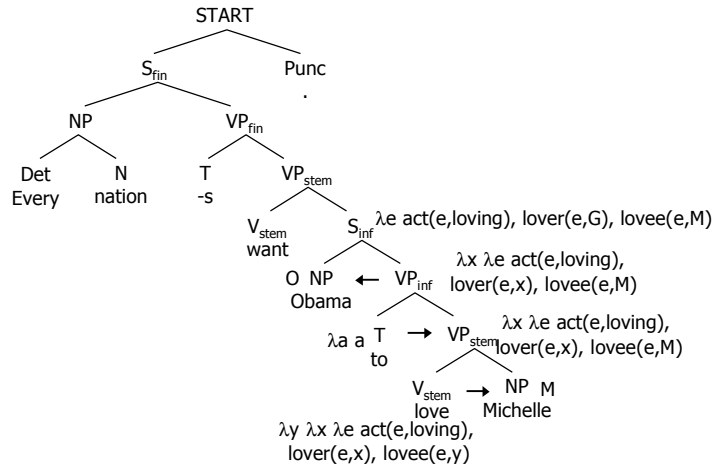


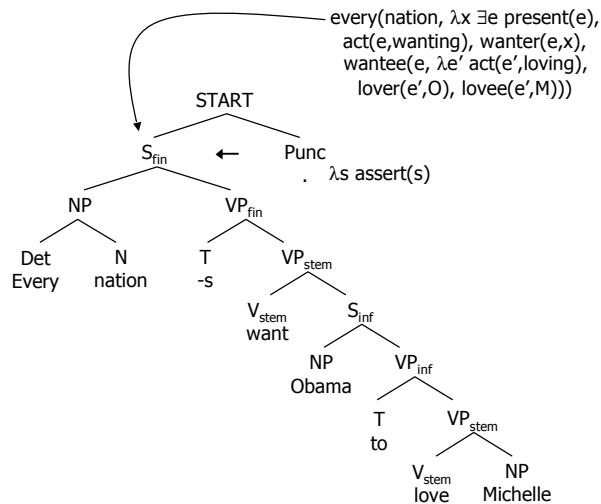
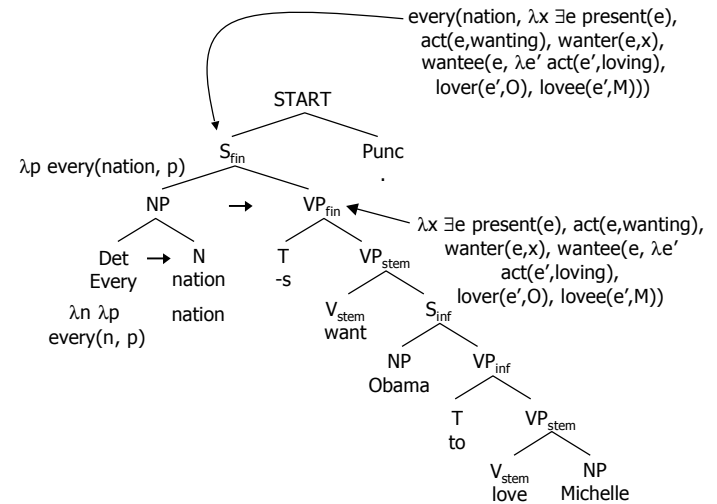
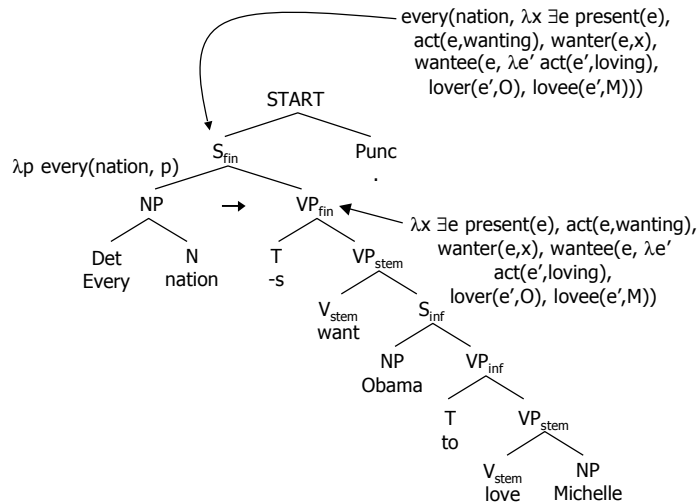
6.863J/9.611J SP11

Now let's try a more complex example, and really handle tense.

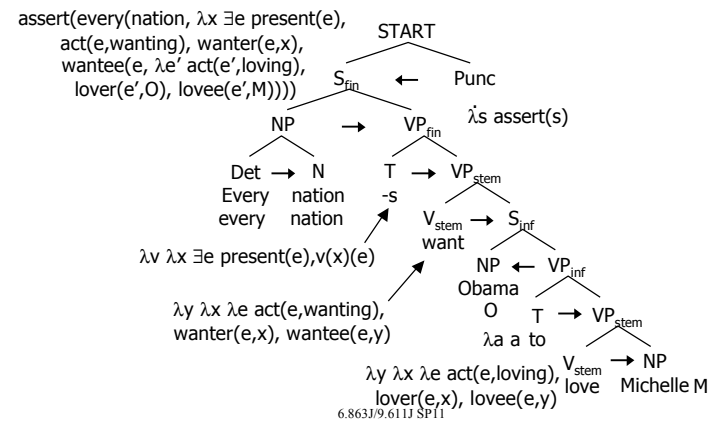








### In Summary: From the Words



## In Summary II: paired syntactic-semantic rules

### Lexicon

*Kathy*, NP : **kathy**

*Fong*, NP : **fong**

*respects*, V :  $\lambda y. \lambda x. \mathbf{respect}(x, y)$

*runs*, V :  $\lambda x. \mathbf{run}(x)$

### Grammar

S :  $\beta(\alpha) \rightarrow$  NP :  $\alpha$  VP :  $\beta$

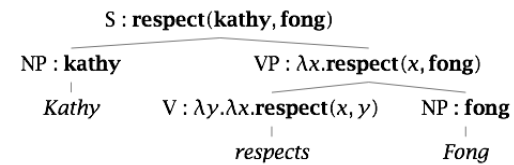
VP :  $\beta(\alpha) \rightarrow$  V :  $\beta$  NP :  $\alpha$

VP :  $\beta \rightarrow$  V :  $\beta$

- Nonterminal : semantic translation rules say how to associate semantic representations with syntactic representations
- In general, head-nonhead syntactic composition corresponds to function application

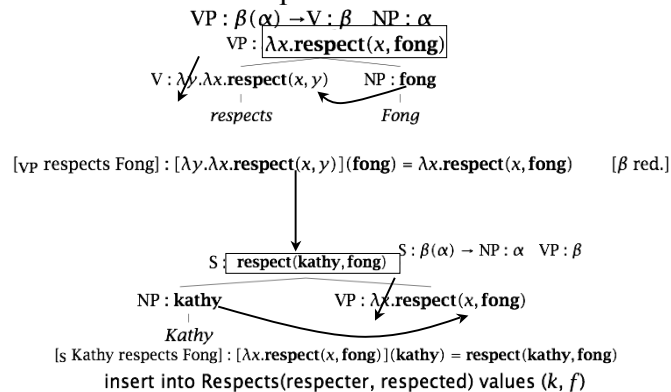
6.863J/9.611J SP11

## Build a decorated tree, with an intermediate ‘semantic’ language



6.863J/9.611J SP11

## Sequence of function applications (beta reductions) tells us how to compose the semantic representation



6.863J/9.611J SP11

## abbreviate some notation...

$\lambda x.(P(x)) \Rightarrow P$   $\eta$  reduction [abstractions can be contracted]

$\lambda y. \lambda x. \mathbf{respect}(x, y)$  (long form)

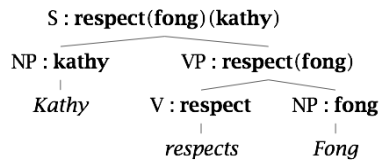
**respect**       $\mathbf{respect}$  is  $\mathbf{Ind} \rightarrow \mathbf{Ind} \rightarrow \mathbf{Bool}$

$\beta, \eta$  long form:  $\lambda x. \mathbf{run}(x), \lambda y. \mathbf{yesterday}(\lambda x. \mathbf{run}(x))(y)$

$\beta, \eta$  normal form: **run, yesterday(run)**

6.863J/9.611J SP11

So we can abbreviate the whole sentence semantics this way...



Which leads to an immediate SQL treatment:

insert into Respects(respecter, respected) values (k, f)

Ok, what's next???

What about prepositions, adjectives, adverbs,...

6.863J/9.611J SP11

New, improved grammar, with semantic augmentation

- $S : \beta(\alpha) \rightarrow NP : \alpha \quad VP : \beta$
- $NP : \beta(\alpha) \rightarrow Det : \beta \quad N' : \alpha$
- $N' : \beta(\alpha) \rightarrow Adj : \beta \quad N' : \alpha$
- $N' : \beta(\alpha) \rightarrow N' : \alpha \quad PP : \beta$
- $N' : \beta \rightarrow N : \beta$
- $VP : \beta(\alpha) \rightarrow V : \beta \quad NP : \alpha$
- $VP : \beta(\gamma)(\alpha) \rightarrow V : \beta \quad NP : \alpha \quad NP : \gamma$
- $VP : \beta(\alpha) \rightarrow VP : \alpha \quad PP : \beta$
- $VP : \beta \rightarrow V : \beta$
- $PP : \beta(\alpha) \rightarrow P : \beta \quad NP : \alpha$

6.863J/9.611J SP11

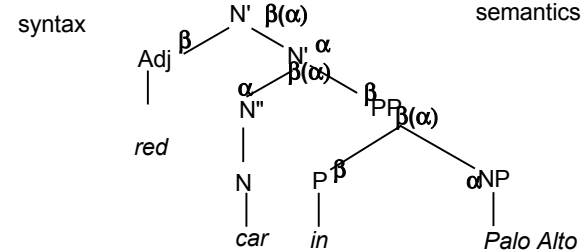
## New lexicon

- Kathy*, NP : **kathy**<sub>Ind</sub>
- Fong*, NP : **fong**<sub>Ind</sub>
- Palo Alto*, NP : **paloalto**<sub>Ind</sub>
- car*, N : **car**<sub>Ind</sub> - Bool
- overpriced*, Adj : **overpriced**<sub>Ind - Bool</sub> - (Ind - Bool) - Bool
- outside*, PP : **outside**<sub>Ind - Bool</sub> - (Ind - Bool) - Bool
- red*, Adj :  $\lambda P. \lambda x. P(x) \wedge \text{red}'(x)$
- in*, P :  $\lambda y. \lambda P. \lambda x. (P(x) \wedge \text{in}'(y)(x))$
- the*, Det :  $\iota$
- a*, Det : **some**<sup>2</sup><sub>Ind - Bool</sub> - (Ind - Bool) - Bool
- runs*, V : **run**<sub>Ind - Bool</sub>
- respects*, V : **respect**<sub>Ind - Bool</sub>
- likes*, V : **like**<sub>Ind - Ind - Bool</sub>
- sees*, V : **see**<sub>Ind - Ind - Bool</sub>
- in'* is **Ind - Ind - Bool**
- $\text{in} \cong \lambda y. \lambda P. \lambda x. (P(x) \wedge \text{in}'(y)(x))$  is **Ind - (Ind - Bool) - (Ind - Bool)**
- red'* is **Ind - Bool**
- $\text{red} \cong \lambda P. (\lambda x. (P(x) \wedge \text{red}'(x)))$  is **(Ind - Bool) - (Ind - Bool)**

6.863J/9.611J SP11

## Let's try it out

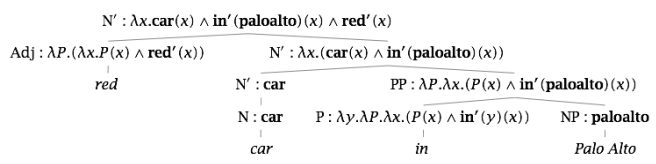
Desired Goal  $\lambda x. \text{car}(x) \wedge \text{in}'(\text{paloalto})(x) \wedge \text{red}'(x)$   
*red car in Palo Alto*



Now add lexical entries and start doing function application...

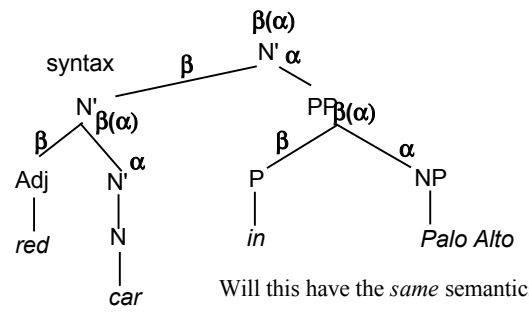
6.863J/9.611J SP11





But... isn't there another parse...?

### Parse #2 - what are *its* semantics?



Will this have the *same* semantics?  
 What does  $\beta$ -reduction construct?