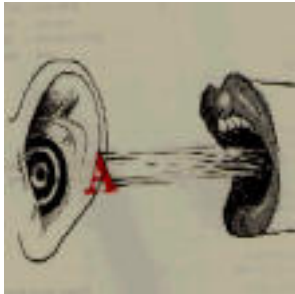


Lecture 11: Semantics



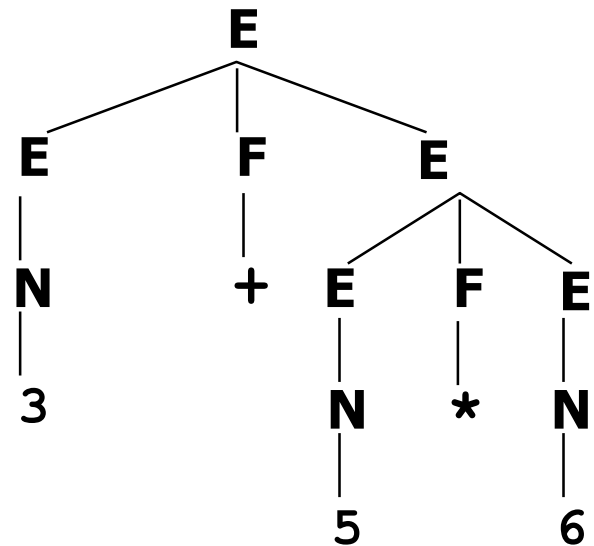
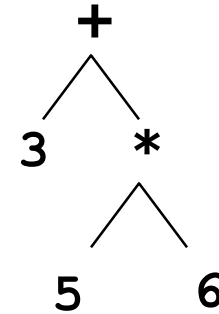
Professor Robert C. Berwick
berwick@csail.mit.edu

The Menu Bar

- Administrivia: Lab 4 out
- What knowledge do we need beyond syntax?
- How should we represent this?
- How can we learn it?

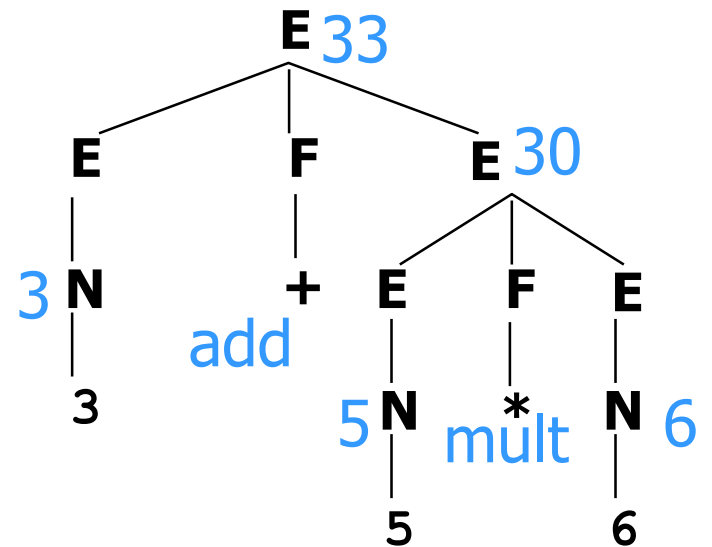
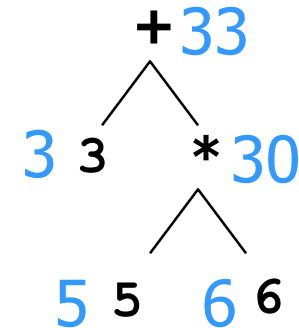
Programming Language Interpreter

- What is the meaning of $3+5*6$?
- First parse it into $3+(5*6)$



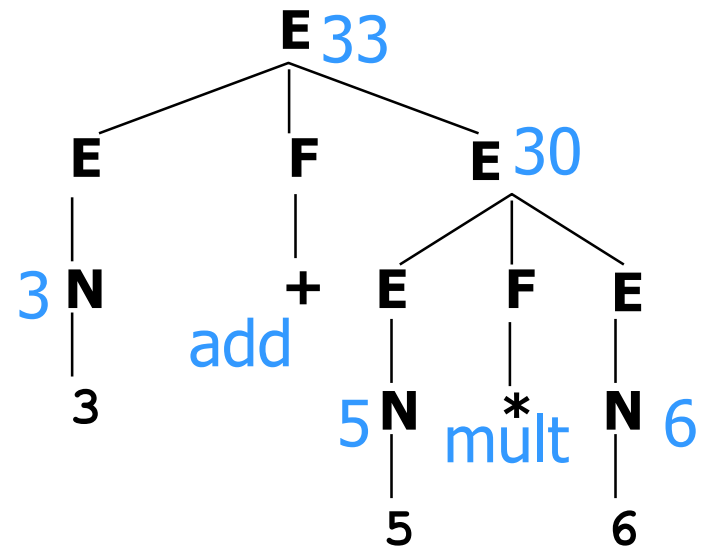
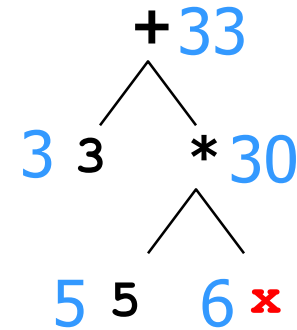
Programming Language Interpreter

- What is meaning of $3+5*6$?
- First parse it into $3+(5*6)$
- Now give a meaning to each node in the tree (bottom-up)



Interpreting in an Environment

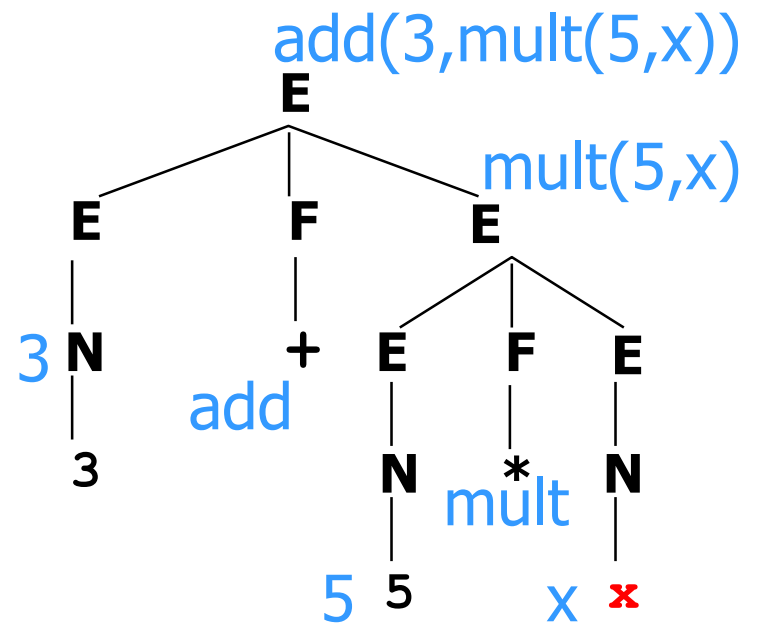
- How about $3+5 * x$?
- Same thing: the meaning of x is found from the environment (it's 6)
- Analogies in language?



Compiling

- How about $3+5 * x$?
- Don't know x at compile time
- “Meaning” at a node is a piece of code, not a number

$5 * (x+1) - 2$ is a different expression that produces *equivalent* code (can be converted to the previous code by optimization)
Analogies in language?



What Counts as Understanding?

some notions

- We understand if we can *respond appropriately*
 - ok for commands, questions (these demand response)
 - “Computer, warp speed 5”
 - “throw axe at dwarf”
 - “put all of my blocks in the red box”
 - imperative programming languages
 - database queries and other questions
- We understand a statement if we can *determine its truth*
 - ok, but if you knew whether it was true, why did anyone bother telling it to you?
 - comparable notion for understanding a noun phrase is to compute what the noun phrase refers to, which might be useful

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?

some notions

- 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 – the definition we'll use!
 - all humans are mortal = $\forall x [\text{human}(x) \Rightarrow \text{mortal}(x)]$
- Assume we have logic-manipulating rules to tell us how to act, draw conclusions, answer questions ...

Plan

- First:
 - Let's look at some sentences and phrases
 - What would be reasonable logical representations for them?
- Then:
 - How can we build those representations?
- Another course (AI):
 - How can we reason with those representations?

Where we stand: How can a computer 'understand' a discourse?

Here's a "discourse". Let's see how to chop it up....

Dolphins are mammals, not fish. They are warm blooded like man, and give birth to one baby called a calf at a time. At birth a bottlenose dolphin calf is about 90-130 cms long and will grow to approx. 4 meters, living up to 40 years. They are highly sociable animals, living in pods which are fairly fluid, with dolphins from other pods interacting with each other from time to time.

At least three parts to ‘semantics’

1. Lexical semantics
2. Logical semantics
3. Discourse semantics

Lexical semantics: nouns & verbs

Dolphins are mammals, not fish. They are warm blooded like man, and give birth to one baby called a calf at a time. At birth a bottlenose dolphin calf is about 90-130 cms long and will grow to approx. 4 meters, living up to 40 years. They are highly sociable animals, living in pods which are fairly fluid, with dolphins from other pods interacting with each other from time to time.

What about 90, pods, ?

Noun

- **S: (n)** [ninety](#), [90](#), **XC** (the cardinal number that is the product of ten and nine)

Adjective

- **S: (adj)** [ninety](#), [90](#), **xc** (being ten more than eighty)

Noun

- **S: (n)** **pod**, [cod](#), [seedcase](#) (the vessel that contains the seeds of a plant (not the seeds themselves))
- **S: (n)** **pod**, [seedpod](#) (a several-seeded dehiscent fruit as e.g. of a leguminous plant)
- **S: (n)** **pod** (a group of aquatic mammals)
- **S: (n)** **pod**, [fuel pod](#) (a detachable container of fuel on an airplane)

Logical semantics: everything becomes a first-order logical operator

Dolphins **are** mammals, **not** fish. They **are** warm blooded like man, **and** give birth to **one** baby called **a** calf at **a** time. At birth **a** bottlenose dolphin calf **is** about 90-130 cms long **and** will grow to approx. 4 meters, living up to 40 years. They **are** highly sociable animals, living in pods **which are** fairly fluid, with dolphins from other pods interacting with each other from time to time.

Discourse semantics: words that tie sentences together (pronouns, ‘anaphora’)

Dolphins are mammals, not fish. **They** are warm blooded like man, and give birth to one baby called a calf at a time. At birth a bottlenose dolphin calf is about 90-130 cms long and will grow to approx. 4 meters, living up to 40 years. **They** are highly sociable animals, living in pods which are fairly fluid, with dolphins from **other** pods interacting with **each other** from time to time.

Question: what words have we missed?

What words are left out so far?

Dolphins are mammals, not fish. They are warm blooded **like** man, and give birth to one baby called a calf **at** a time. **At** birth a bottlenose dolphin calf is **about** 90-130 cms long and **will** grow to **approx.** 4 meters, living **up to** 40 years. They are **highly** sociable animals, living **in** pods which are **fairly** fluid, **with** dolphins **from** other pods interacting **with** each other **from** time to time.

Some of these are obviously ‘logical’...

What are we left with?

Dolphins are mammals, not fish. They are warm blooded **like** man, and give birth to one baby called a calf **at** a time. **At** birth a bottlenose dolphin calf is **about** 90-130 cms long and **will** grow to **approx.** 4 meters, living **up to** 40 years. They are **highly** sociable animals, living **in** pods which are **fairly** fluid, **with** dolphins **from** other pods interacting **with** each other **from** time to time.

Red items = in wordnet, but need translation to logic;

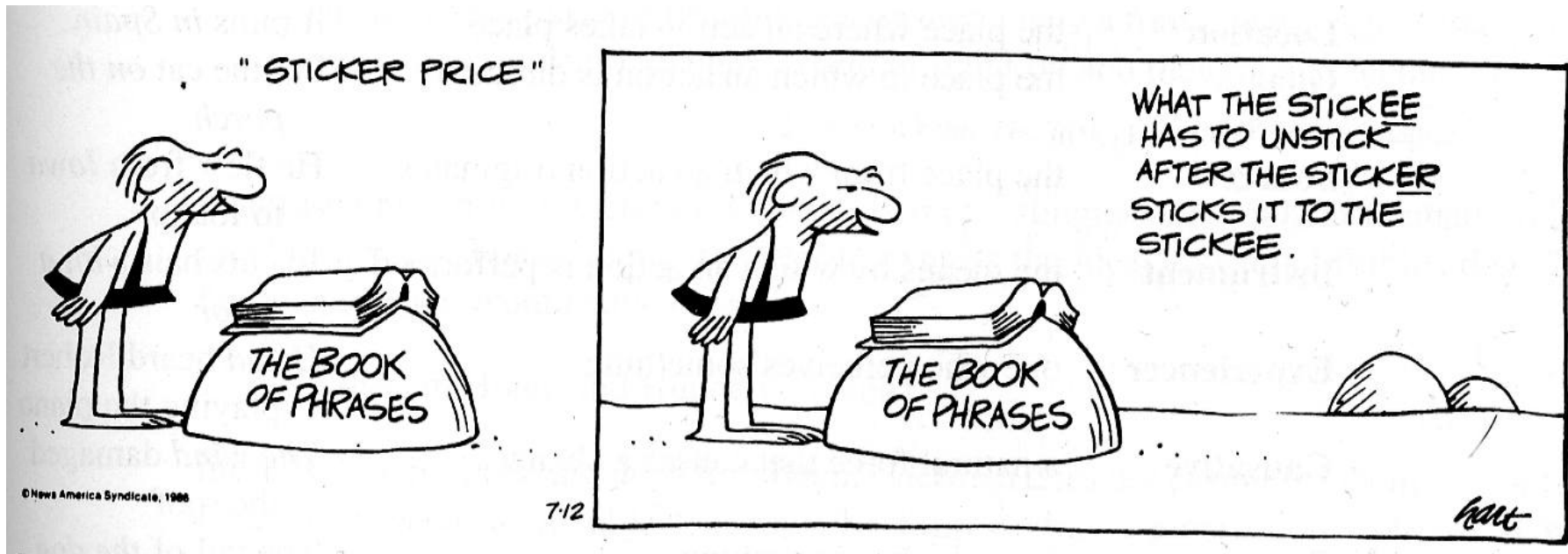
Blue items = require logical translation ‘from time to time’

Today, we’ll focus on the logical semantics

Logical semantics: everything that becomes a first-order logical operator, including quantifiers

Dolphins **are** mammals, **not** fish. They **are** warm blooded like man, **and** give birth to **one** baby called **a** calf at **a** time. At birth **a** bottlenose dolphin calf **is about** 90-130 cms long **and** will grow to **approx.** 4 meters, living up to 40 years. They **are highly** sociable animals, living in pods **which are** fairly fluid, with dolphins from other pods interacting with each other **from** time to time.

The key information is in the argument structure of verbs



Logic: Some Preliminaries

Three major kinds of objects

Booleans

- Roughly, the semantic values of sentences

Entities

- Values of NPs, e.g., objects like this slide
- Maybe also other types of entities, like times

Functions of various types

- A function returning a boolean is called a “predicate”
– e.g., `frog(x)`, `green(x)`
- Functions might return other functions!
- Function might take other functions as arguments!

We use lambda calculus as our computational method - Why?

1. Lets us 'impedance match' syntactic rep to semantic rep when they don't line up (eg, $S \rightarrow NP VP$, not always head-nonhead/function-arg form)
2. Lets us delay finding actual values by using dummy variable names and waiting until we've processed more of the sentence
3. Lets us turn multiple argument functions into a sequence of 1-argument functions (more later)

Lambda calculus & lambda terms

- Key to building logical semantic representations
- Extension of first order predicate calculus allowing us to bind variables using operator λ
- Occurrences of variables bound by λ are place holders 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 representation – a construction kit, it's the Elmer's glue

Logic: Lambda Terms

- Lambda terms:
 - A way of writing “anonymous functions”
 - No function header or function name
 - But defines the key thing: **behavior** of the function
 - Just as we can talk about 3 without naming it “x”
 - Let `square = λp p*p`
 - Equivalent to `int square(p) { return p*p; }`
 - But we can talk about `λp p*p` *without naming it*
 - Format of a lambda term: `λ variable expression`

Logic: Lambda Terms

- Lambda terms:
 - Let $\text{square} = \lambda p p * p$
 - Then $\text{square}(3) = (\lambda p p * p)(3) = 3 * 3$
 - **Note: $\text{square}(x)$ isn't a function! It's just the value $x * x$.**
 - But $\lambda x \text{square}(x) = \lambda x x * x = \lambda p p * p = \text{square}$
(proving that these functions are equal – and indeed they are,
as they act the same on all arguments: what is $(\lambda x \text{square}(x))(y)$?)
 - Let $\text{even} = \lambda p (p \bmod 2 == 0)$ a predicate: returns true/false
 - $\text{even}(x)$ is true if x is even
 - How about $\text{even}(\text{square}(x))$?
 - $\lambda x \text{even}(\text{square}(x))$ is true of numbers with even squares
 - Just apply rules to get $\lambda x (\text{even}(x * x)) = \lambda x (x * x \bmod 2 == 0)$
 - This happens to denote the same predicate as even does

Function application works by ‘beta reduction’

$$\lambda x P(\dots, x, \dots)Z \Rightarrow P(\dots, Z, \dots)$$

e.g., $\lambda x \text{ runs}(x)@Kathy \Rightarrow \text{runs}(Kathy)$

- Remove lambda, replace instances of bound variable by Z

Logic: Multiple Arguments

- All lambda terms have one argument
- But we can fake multiple arguments ...
- Suppose we want to write $\text{times}(5,6)$
- Remember: square can be written as $\lambda x \text{ square}(x)$
- Similarly, times is equivalent to $\lambda x \lambda y \text{ times}(x,y)$
- **Claim that $\text{times}(5)(6)$ means same as $\text{times}(5,6)$**
 - $\text{times}(5) = (\lambda x \lambda y \text{ times}(x,y)) (5) = \lambda y \text{ times}(5,y)$
 - If this function weren't anonymous, what would we call it?
 - $\text{times}(5)(6) = (\lambda y \text{ times}(5,y))(6) = \text{times}(5,6)$

Logic: Multiple Arguments

- All lambda terms have one argument
- But we can fake multiple arguments ...
- **Claim that $\text{times}(5)(6)$ means same as $\text{times}(5,6)$**
 - $\text{times}(5) = (\lambda x \lambda y \text{times}(x,y)) (5) = \lambda y \text{times}(5,y)$
 - If this function weren't anonymous, what would we call it?
 - $\text{times}(5)(6) = (\lambda y \text{times}(5,y))(6) = \text{times}(5,6)$
- **So we can always get away with 1-arg functions ...**
 - ... which might return a function to take the next argument.
Whoa.
 - We'll still allow $\text{times}(x,y)$ as syntactic sugar, though

Grounding out

- So what does **times** actually mean???
- How do we get from **times(5,6)** to **30** ?
 - Whether **times(5,6) = 30** depends on whether symbol **times** actually denotes the multiplication function!
- Well, maybe **times** was defined as another lambda term, so substitute to get **times(5,6) = (blah blah blah)(5)(6)**
- But we can't keep doing substitutions forever!
 - Eventually we have to ground out in a **primitive term**
 - Primitive terms are bound to object code
- Maybe **times(5,6)** just executes a multiplication function
- Just what is executed by **loves(john, mary)** ?

Logic: Interesting Constants

- Thus, have “constants” that name some of the entities and functions (e.g., **times**):
 - **BarackObama** - an entity
 - **red** – a predicate on entities
 - holds of just the red entities: $\text{red}(x)$ is true if x is red!
 - **loves** – a predicate on 2 entities
 - **loves(BarackObama,MichelleObama)**
 - *Question:* What does **loves(MichelleObama)** denote?
- Constants used to define meanings of words
- Meanings of phrases will be built from the constants

Logic: Interesting Constants

- **most** – a predicate on 2 predicates on entities
 - **most(pig, big)** = “most pigs are big”
 - Equivalently, **most(λx pig(x), λx big(x))**
 - returns true if most of the things satisfying the first predicate also satisfy the second predicate (is this ok?? We will see more about ‘most’ a bit later...)
- similarly for other quantifiers
 - **all(pig, big)** (equivalent to $\forall x$ pig(x) \Rightarrow big(x))
 - **exists(pig, big)** (equivalent to $\exists x$ pig(x) AND big(x))
 - can even build complex quantifiers from English phrases:
 - “between 12 and 75”; “a majority of”; “all but the smallest 2” (again, this is more subtle... as we shall see)

A reasonable representation?

- *Gilly swallowed a goldfish*
- First attempt: `swallowed(Gilly, goldfish)`
- Returns true or false. Analogous to:
 - `prime(17)`
 - `equal(4,2+2)`
 - `loves(BarackObama, MichelleObama)`
 - `swallowed(Gilly, Jilly)`
- ... or is it analogous??? What's wrong with this?

A reasonable representation?

- *Gilly swallowed a goldfish*
 - First attempt: `swallowed(Gilly, goldfish)`
- But we're not paying attention to *a*!
- *goldfish* isn't the name of a unique object the way *Gilly* is
- In particular, don't want
Gilly swallowed a goldfish and Milly swallowed a goldfish
to translate as:

`swallowed(Gilly, goldfish) AND swallowed(Milly, goldfish)`

since probably not the same goldfish ...

Use a Quantifier

- *Gilly swallowed a goldfish*
 - First attempt: `swallowed(Gilly, goldfish)`
- Better: $\exists g \text{ goldfish}(g) \text{ AND } \text{swallowed}(\text{Gilly}, g)$
- Or using one of our quantifier predicates:
 - $\text{exists}(\lambda g \text{ goldfish}(g), \lambda g \text{ swallowed}(\text{Gilly}, g))$
 - Equivalently: $\text{exists}(\text{goldfish}, \text{swallowed}(\text{Gilly}))$
 - “In the set of goldfish there exists one swallowed by Gilly”
- Here `goldfish` is a predicate on entities
 - This is the same semantic type as `red`
 - But *goldfish* is noun and *red* is adjective .. #@!?

Tense

- *Gilly swallowed a goldfish*
 - Previous attempt: $\text{exists}(\text{goldfish}, \lambda g \text{ swallowed}(\text{Gilly}, g))$
- Improve to use tense:
 - Instead of the 2-arg predicate $\text{swallowed}(\text{Gilly}, g)$ try a 3-arg version $\text{swallow}(t, \text{Gilly}, g)$ where t is a time
 - Now we can write:
 $\exists t \text{ past}(t) \text{ AND } \text{exists}(\text{goldfish}, \lambda g \text{ swallow}(t, \text{Gilly}, g))$
 - “There was some time in the past such that a goldfish was among the objects swallowed by Gilly at that time”
 - We’ll see later on that tense has some interesting constraints across languages...with implications about *how* to represent it (For example: *John will study Turkish when Bill arrived*)

(Simplify Notation)

- *Gilly swallowed a goldfish*
 - Previous attempt: `exists(goldfish, swallowed(Gilly))`
- Improve to use tense:
 - Instead of the 2-arg predicate `swallowed(Gilly,g)` try a 3-arg version `swallow(t,Gilly,g)`
 - Now we can write:
`∃t past(t) AND exists(goldfish, swallow(t,Gilly))`
 - “There was some time in the past such that a goldfish was among the objects swallowed by Gilly at that time”

Event Properties

- *Gilly swallowed a goldfish*
 - Previous: $\exists t \text{ past}(t) \text{ AND exists}(\text{goldfish}, \text{swallow}(t, \text{Gilly}))$
- Why stop at time? An event has other properties:
 - [Gilly] swallowed [a goldfish] [on a dare] [in a telephone booth] [with 30 other freshmen] [after many bottles of vodka had been consumed].
 - Specifies who what why when ...
- Replace time variable t with an event variable e
 - $\exists e \text{ past}(e), \text{act}(e, \text{swallowing}), \text{swallower}(e, \text{Gilly}), \text{exists}(\text{goldfish}, \text{swallowee}(e)), \text{exists}(\text{booth}, \text{location}(e)), \dots$
 - As with probability notation, a comma represents AND
 - Could define *past* as $\lambda e \exists t \text{ before}(t, \text{now}), \text{ended-at}(e, t)$

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

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?

Quantifier Order

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

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”

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, ...*

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

Control

- *Willy wants Lilly to get married*
 - $\exists e \text{ present}(e), \text{act}(e, \text{wanting}), \text{wanter}(e, \text{Willy}), \text{wantee}(e, \lambda f [\text{act}(f, \text{marriage}), \text{marrier}(f, \text{Lilly})])$
- *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.”

Nouns and Their Modifiers

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

Nouns and Their Modifiers

- *the goldfish that Gilly swallowed*
- *every goldfish that Gilly swallowed*
- *three goldfish that Gilly swallowed*

λg [goldfish(g), swallowed(Gilly, g)]

like an adjective!

- *three swallowed-by-Gilly goldfish*

Or for real: λg [goldfish(g), $\exists e$ [past(e), act(e,swallowing),
swallower(e,Gilly), swallowee(e,g)]]

Adverbs

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

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.

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???

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

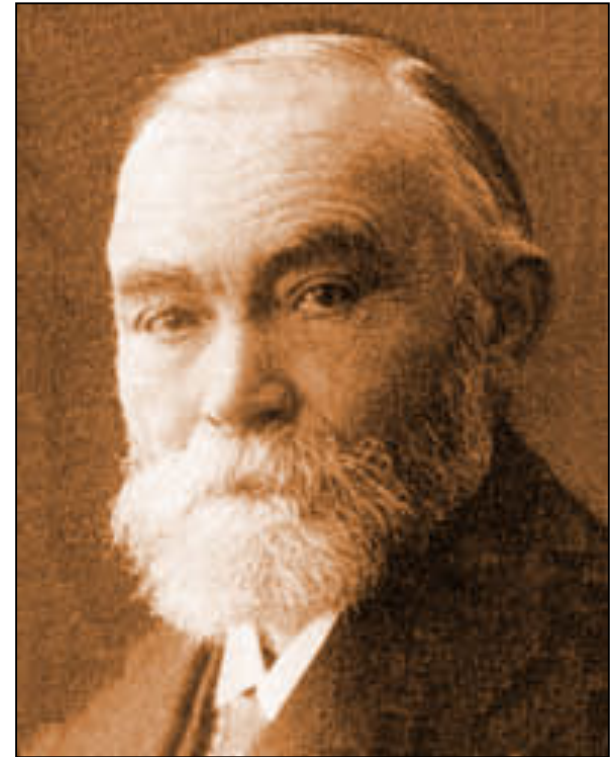
The Principles

- [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

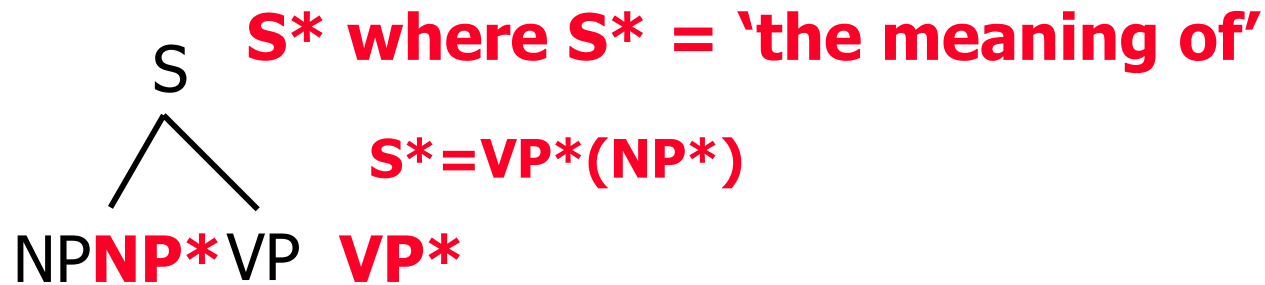


Context-free semantics: associate interpretation rule with each context-free rule

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

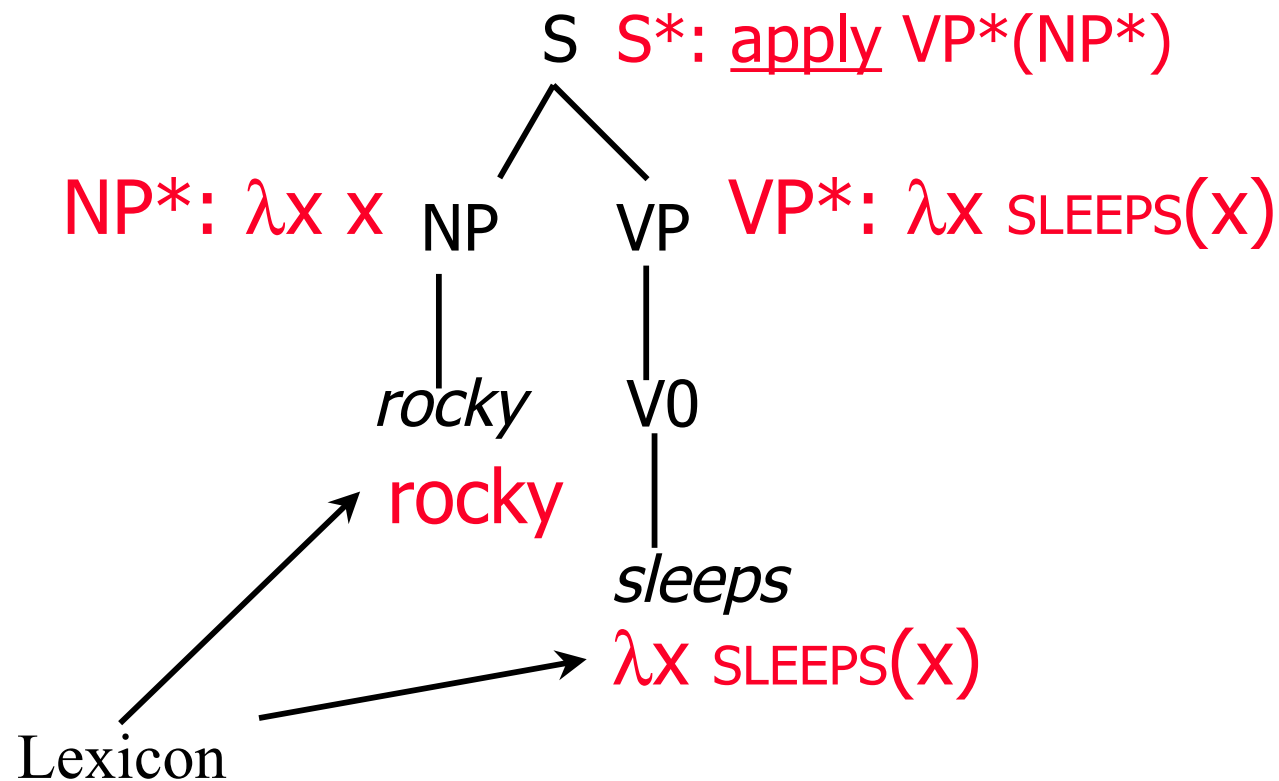
The master principles

- 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

Rule-to-rule principle: decorate syntax tree with lambda expressions



6.001 to the rescue

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

$\lambda x \text{ SLEEPS}(x)$

- When this lambda abstraction - a function - is applied to the argument 'Poirot', as usual this binds the variable x :

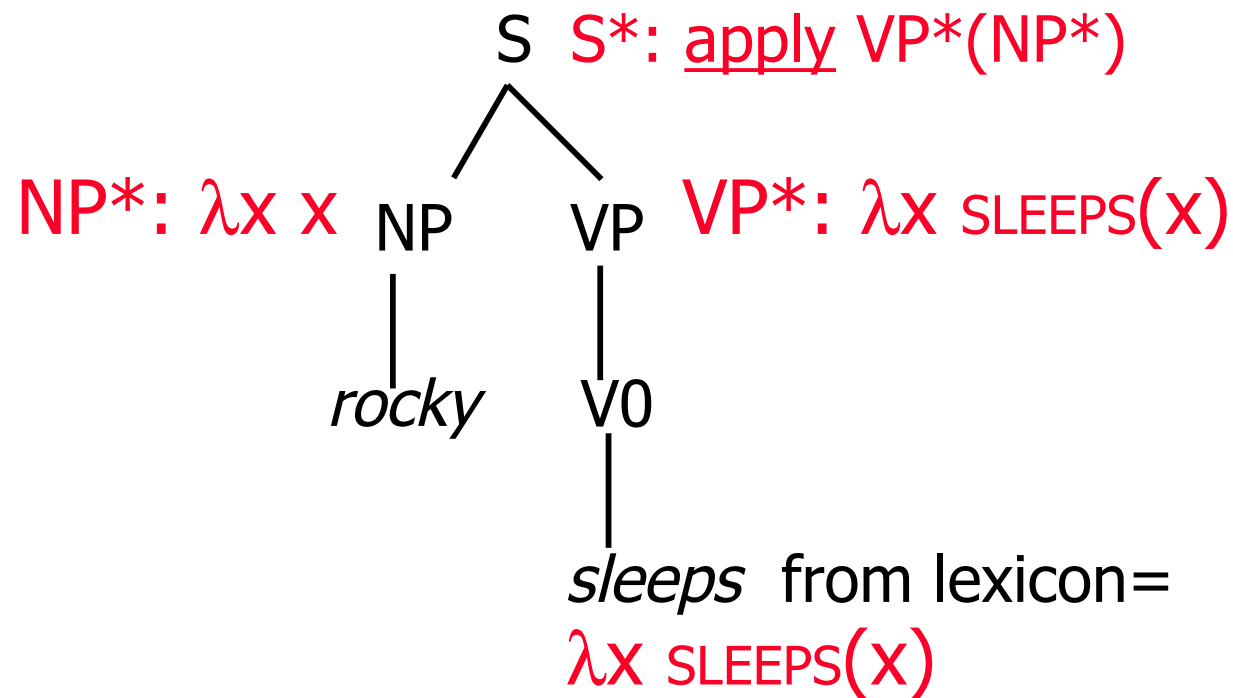
$\lambda x \text{ SLEEPS}(x).\text{rocky} \rightarrow \text{SLEEPS}(\text{rocky})$

- The lambda variable acts as a placeholder for missing info
- The “**rockyt**” denotes the substitution
- Alternatively, to make this more visible, we write:

$\lambda x \text{ SLEEPS}(x)\text{@rocky} \rightarrow \text{SLEEPS}(\text{rocky})$

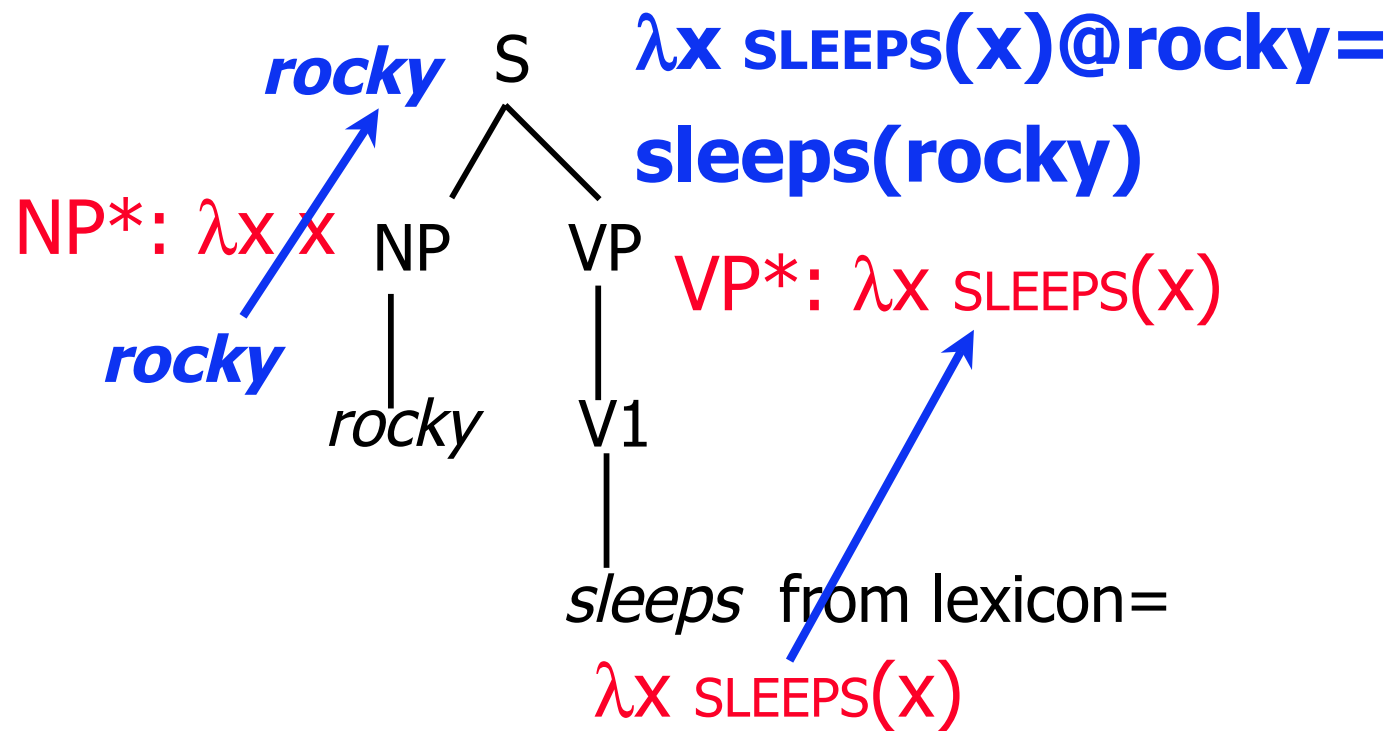
- Where the “**@**” denotes function application
- This process of substitution is also called beta reduction

Rule-to-rule principle: decorate syntax tree with



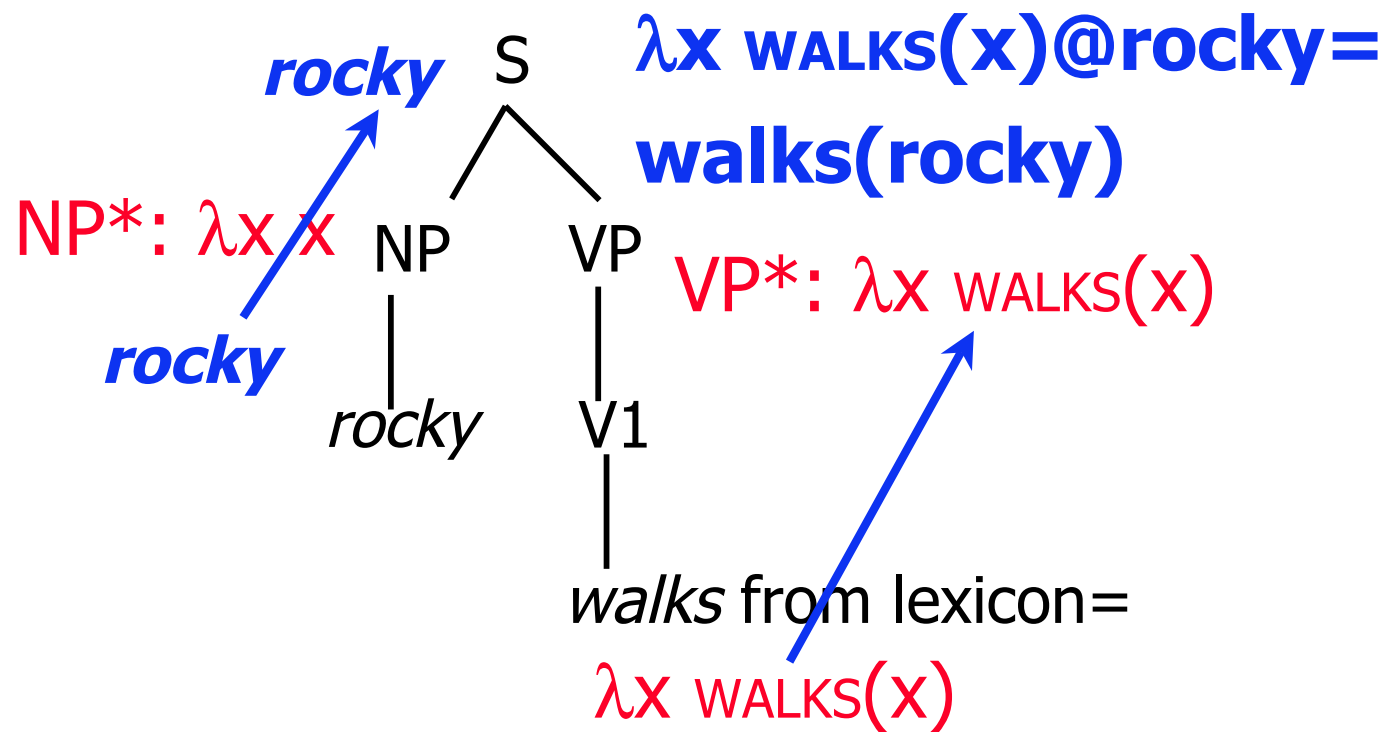
Doing the beta reductions...

S^* : apply $VP^*(NP^*) \rightarrow$



What about 'rocky walks'?

S^* : apply $VP^*(NP^*) \rightarrow$



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

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)`

DB – the ‘truth’ as SQL

Rocky likes Bullwinkle →

Insert into Like (liker, 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.liker = r
and Like.likee = b**

Example of what we might do: text understanding via question-answering

```
athena>(top-level)
```

```
Shall I clear the database? (y or n) y
```

```
sem-interpret>John saw Mary in the park
```

```
OK.
```

```
sem-interpret>Where did John see Mary
```

```
IN THE PARK.
```

```
sem-interpret>John gave Fido to Mary
```

```
OK.
```

```
sem-interpret>Who gave John Fido
```

```
I DON'T KNOW
```

```
sem-interpret>Who gave Mary Fido
```

```
JOHN
```

```
sem-interpret >John saw Fido
```

```
OK.
```

```
sem-interpret>Who did John see
```

```
FIDO AND MARY
```

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.

Compositional Semantics

assert(every(nation, $\lambda x \exists e$ present(e),
 act(e,wanting), wanter(e,x),
 wantee(e, $\lambda e'$ act(e',loving),
 lover(e',O), lovee(e',M))))

