
Foundations of Natural Language Processing

Lecture 18

Compositional Semantics

Alex Lascarides



Meaning

Last Time: Syntactic parsing

Now: How can syntax inform **meaning**?

- Desiderata for formal semantic representations of sentences
- Method for constructing semantic representations from a syntax tree.

Meaning

- One grand goal of artificial intelligence is to *understand* what people mean when they talk.
- But how do we know if we succeeded?

Literal Meaning

Semantics is concerned with how expressions relate to 'the world'. This includes both their

- **truth conditions and denotation** (literal meaning)
- **connotation** (other associations)

For now, we're interested only in literal meaning.

Eliza

Joseph Weizenbaum wrote the computer program **Eliza** in 1969 to demonstrate how easily people can be fooled into thinking that machines have some deep understanding.

young woman: Men are all alike.

eliza: In what way?

young woman: They're always bugging us about something specific or other.

eliza: Can you think of a specific example?

young woman: Well, my boyfriend made me come here.

eliza: Your boyfriend made you come here?

young woman: He says I'm depressed much of the time.

eliza: I am sorry to hear you are depressed.

young woman: It's true. I'm unhappy.

eliza: Do you think coming here will help you not to be unhappy?

Online demo: <https://psych.fullerton.edu/mbirnbaum/psych101/eliza.htm>

Large Language Models

- Huge models trained ($\approx 90\text{B}$ parameters) on trillions of words
- Tasked with predicting the next word, given context.
- Big Data leads to massive sensitivity of context.
- Result is fluent, coherent,
- But no predictable relationship to truth about the world
- No underlying intent (even if the user is fooled to think there is).

Large Language Models: Hallucination

user: who is the sole survivor of the Titanic

ChatGPT: The sole survivor of the Titanic disaster was Charles Joughin.
He was a baker. . .

user: Can you provide sources for this answer?

ChatGPT: I apologise for any confusion, but there has been a misunderstanding.
Charles Joughin was not the sole survivor. . .

- What if you didn't ask the question about sources?
- ChatGPT makes up citations and sources in its answers anyway
- Regularly gets facts in requested biographies wrong. Defamation?

What is meaning? What is understanding?

- These are deep philosophical questions
- NLP usually takes a more pragmatic view: can the computer behave as though it understands (in order to do what we want)?
 - Dialogue systems (e.g., Eliza)
 - Machine translation
 - Question answering
- What issues will we face in building such systems?

A Concrete Goal

- We would like to build
 - a machine that answers questions in natural language.
 - may have access to knowledge bases
 - may have access to vast quantities of English text
- Basically, think Alexa!
- This is typically called **Question Answering**

Semantics

- To build our QA system we will need to deal with issues in **semantics**, i.e., meaning.
- Sentential semantics: how word meanings combine (study now)
 - Who did what to whom; when, how, why. . .
 John loves Mary \neq Mary loves John
 \Rightarrow Someone loves Mary
- Lexical semantics: the meanings of individual words (study after that)
E.g., John is male, Mary is female,
loves is more closely related to like than to sees, antonym of hate. . .

What we've learned so far about sentential syntax

- It captures linguistic generalisations about grammaticality (*substitutability*)
- It generates an unbounded set of grammatical sentences via a finite lexicon and finite rules (*recursion*)
- We can induce probabilistic grammars from a treebank, and so tackle (pervasive) syntactic ambiguity.

- Sentential syntax reveals information about sentence meaning
John loves Mary \mapsto *love(j, m)*
Mary loves John \mapsto *love(m, j)*
 - Decisions about how to resolve syntactic ambiguity are tied up with decisions about (intended) meaning.
 - Syntactic ambiguity (almost) always yields a semantic ambiguity.
 - Resolving syntactic ambiguities does *not*, however, resolve all semantic ambiguities
 - word sense, semantic scope, anaphoric expressions
(*all to be studied later in this course*)
- so reasoning about **context** is also very important
(*also studied later in this course*)

What we'll study now. . .

- Principle of **Compositionality**
- Exploit compositionality to augment a grammar with a **semantic component**, which *deterministically derives* the **logical form** of a sentence from its syntax tree.

Desiderata for (Literal) Semantic Representations

The semantic representation should:

- be **unambiguous**
(> 1 semantic representation for *I made her duck* etc)
- support **automated inference**
- be **verifiable**: determine if the sentence is *true* with respect to a model of the world.

Answer: First order logic

An Aside: Logical vs. Commonsense inference

For now:

- John buttered toast at midnight on the lawn \Rightarrow
Someone buttered toast, Someone did something at midnight

For later:

- The purchase of Houston-based LexCorp by BMI for \$2B prompted widespread sell-offs by traders \Rightarrow
BMI acquired an American company (from RTE)
- John buttered toast at midnight on the lawn \Rightarrow
Some food preparation took place in the dark, with the cook standing on grass.

Why FoL and not Propositional Logic?

Fred eats lentils or he eats rice. ($P \vee Q$)

Fred eats rice or John eats rice ($P \vee R$)

- Doesn't capture the internal structure of the proposition *Fred ate rice* (e.g. how its meaning is derived from that of "Fred", "ate", "rice").
- We're unable to express important relationships between, e.g.
 - Everyone eats rice \vdash Someone eats rice, Everyone eats something.
 - Fred eats rice \vdash Someone eats rice

- Fred ate rice: $eat(fred, rice)$ (i)
- Everyone ate rice: $\forall x. eat(x, rice)$ (ii)
- Someone ate rice: $\exists x. eat(x, rice)$ (iii)
- Every dog had a bone: $\forall x(dog(x) \rightarrow \exists y(bone(y) \wedge have(x, y)))$ (iv)
- $\exists y(bone(y) \wedge \forall x.(dog(x) \rightarrow have(x, y)))$ (v)

(ii) entails (i) and (iii); (i) entails (iii); (v) entails (iv)!

Adding tense and modifiers: Davidsonian Semantics

Introducing an event argument e to 'action' predicates is very useful:

Tense: Fred ate rice: $\exists e(\text{eat}(e, \text{fred}, \text{rice}) \wedge e \prec n)$

Modifiers: Fred ate rice with a fork at midnight:

$$\begin{aligned} &\exists e(\text{eat}(e, \text{fred}, \text{rice}) \wedge e \prec n \wedge \\ &\quad \exists x(\text{with}(e, x) \wedge \text{fork}(x)) \wedge \\ &\quad \text{at}(e, \text{midnight}) \end{aligned}$$

Note how the second sentence entails the first via \wedge -elimination!

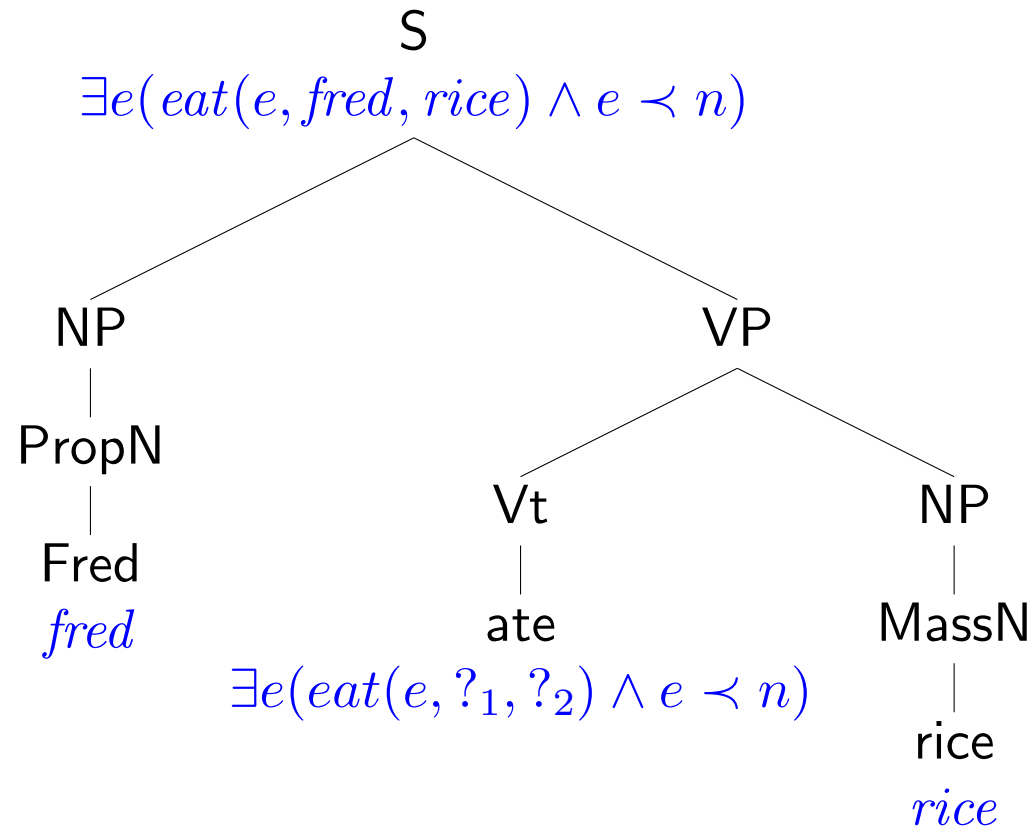
Compositionality

- **Compositionality:** The meaning of a complex expression is a function of the meaning of its parts and of the rules by which they are combined.
- So you can build a **logical form** of a sentence by specifying:

Lexical meanings: Associate each word in the lexicon with a FoL expression.

Composition rules: Augmenting each syntax rule in a CFG with instructions for composing the FoL expressions on the RHS into a FoL expression for the LHS.

What we're aiming for



- How do we get the bits to combine?
- What are the LFs of the intermediate nodes?

Lambda Calculus and Beta Reduction

Allows us to work with 'partially constructed' formulae!

- If φ is a well-formed FoL expression and x is a variable, then $\lambda x\varphi$ is a well-formed FoL expression. It's a **function**, known as a **λ -term**.
- λ -terms have interesting semantics, but they also offer a way of substituting (free) variables in an FoL expression with values.

$$\lambda x\varphi(a) = \varphi[x/a]$$

- Creating a function $\lambda x\varphi$ from an expression φ is called **Lambda (λ) abstraction**
Function application is called **Beta (β) reduction**.

Example:

- $\lambda y\lambda x(\exists e(\text{eat}(e, x, y) \wedge e \prec n))(\text{rice})$ becomes
 $\lambda x(\exists e(\text{eat}(e, x, \text{rice}) \wedge e \prec n))$

Introducing variables corresponding to properties, relations. . .

- If we introduces variables of 'higher type' then we can substitute variables corresponding to properties, relations etc with values that can be λ -terms!!
- $\lambda P.P(\text{fred})$:
the properties of Fred (man, tall,. . .)
 $\lambda P.P(\text{fred})(\text{man})$ becomes $\text{man}(\text{fred})$

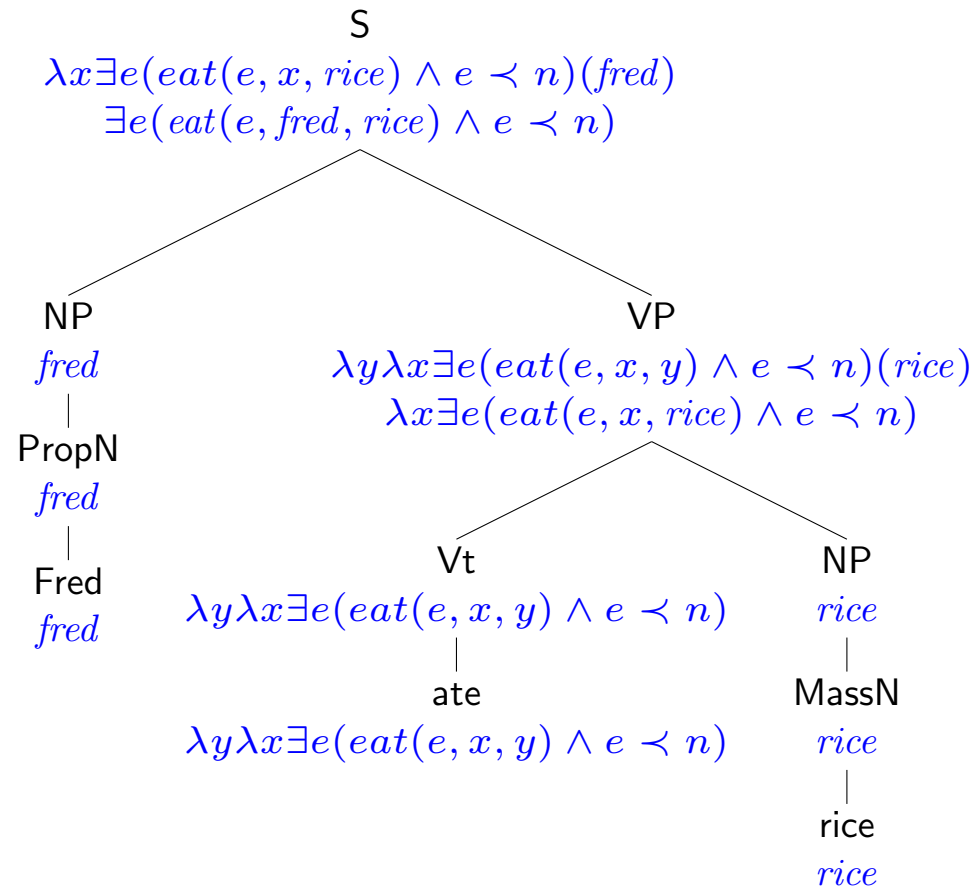
An example where the argument is a λ -term:

- $\lambda P(P(\text{fred}))(\lambda x(\exists e(\text{eat}(e, x, \text{rice}) \wedge e \prec n)))$ becomes
 $\lambda x(\exists e(\text{eat}(e, x, \text{rice}) \wedge e \prec n))(\text{fred})$ becomes
 $\exists e(\text{eat}(e, \text{fred}, \text{rice}) \wedge e \prec n)$

Example Composition for Fred ate rice

But we'll see in a minute why it's problematic. . .

. . . and why λ -abstraction on higher types provides a solution!



The Grammar that generates that tree

$S \rightarrow NP VP$	$VP.Sem(NP.Sem)$	(Sentences)	
$NP \rightarrow MassN$	$MassN.Sem$ $PropN$	$PropN.Sem$	(Noun phrases)
$VP \rightarrow Vi$	$Vi.Sem$ $Vt NP$	$Vt.Sem(NP.Sem)$	(Verb phrases)
$PropN \rightarrow Fred$	$fred$ Jo	jo . . .	(Proper nouns)
$MassN \rightarrow rice$	$rice$ $wood$	$wood$. . .	(Mass nouns)
$Vi \rightarrow talked$	$\lambda x \exists e (talk(e, x) \wedge e \prec n)$. . .	(Intransitive verbs)
$Vt \rightarrow ate$	$\lambda y \lambda x . \exists e (eat(e, x, y) \wedge e \prec n)$. . .	(Transitive verbs)

Observations:

- λ -term for Vt ensures NP values are in right positions to predicate *eat*
- Rules with two daughters specify in semantics which daughter is the functor and which the argument
 - S rule: VP is the functor.
 - Transitive VP rule: Vt is the functor.
- Unary rules ‘pass up’ the semantics from the daughter.

Problematic!

Every man ate rice: $\forall x(\text{man}(x) \rightarrow \exists e(\text{eat}(e, x, \text{rice}) \wedge e \prec n))$

Breaking it down:

- What is the meaning of *Every man* anyway?

$\forall x(\text{man}(x) \rightarrow Q(x))$

- If so, the subject NP needs to be:

$\lambda Q \forall x(\text{man}(x) \rightarrow Q(x))$

- But in our grammar we had the VP as the functor:

$S \rightarrow \text{NP VP } VP.Sem(\text{NP.Sem})$

- $\lambda z \exists e(\text{eat}(e, z, \text{rice}) \wedge e \prec n)(\lambda Q \forall x(\text{man}(x) \rightarrow Q(x)))$ becomes

$\lambda z \exists e(\text{eat}(e, \lambda Q \forall x(\text{man}(x) \rightarrow Q(x)), \text{rice}) \wedge e \prec n)$

- That's not even syntactically well-formed!!

Solution

Make NP the functor and VP the argument.

$$S \rightarrow NP \ VP \ NP.Sem(VP.Sem)$$

$$\begin{aligned} & \lambda Q \forall x (man(x) \rightarrow Q(x)) (\lambda z \exists e (eat(e, z, rice) \wedge e \prec n)) \\ & \forall x (man(x) \rightarrow \lambda z \exists e (eat(e, z, rice) \wedge e \prec n))(x) \\ & \forall x (man(x) \rightarrow \exists e (eat(e, x, rice) \wedge e \prec n)) \end{aligned}$$

But this means NPs must all look like this: $\lambda P.P(x)$.

Fred $\mapsto \lambda P.P(fred)$ etc.

Now a problem with transitive verbs!!

ate

every grape:

$$\lambda y \lambda z \exists e (eat(e, z, y) \wedge e \prec n) \quad \lambda Q \forall x (grape(x) \rightarrow Q(x))$$

NP.Sem(Vt.Sem) is ill formed!

$\lambda Q \forall x (grape(x) \rightarrow Q(x)) (\lambda y \lambda z \exists e (eat(e, z, y) \wedge e \prec n))$ becomes

$\forall x (grape(x) \rightarrow \lambda y \lambda z \exists e (eat(e, z, y) \wedge e \prec n)(x))$ becomes

$\forall x (grape(x) \rightarrow \lambda z \exists e (eat(e, z, x) \wedge e \prec n))$

ill-formed!

It should be: $\lambda z \forall x (grape(x) \rightarrow \exists e (eat(e, z, x) \wedge e \prec n))$

Type Raising to the rescue again

VP \rightarrow Vt NP $Vt.Sem(NP.Sem)$
Vt \rightarrow ate $\lambda R.\lambda z.R(\lambda y.\exists e(eat(e, z, y) \wedge e \prec n))$

ate every grape:

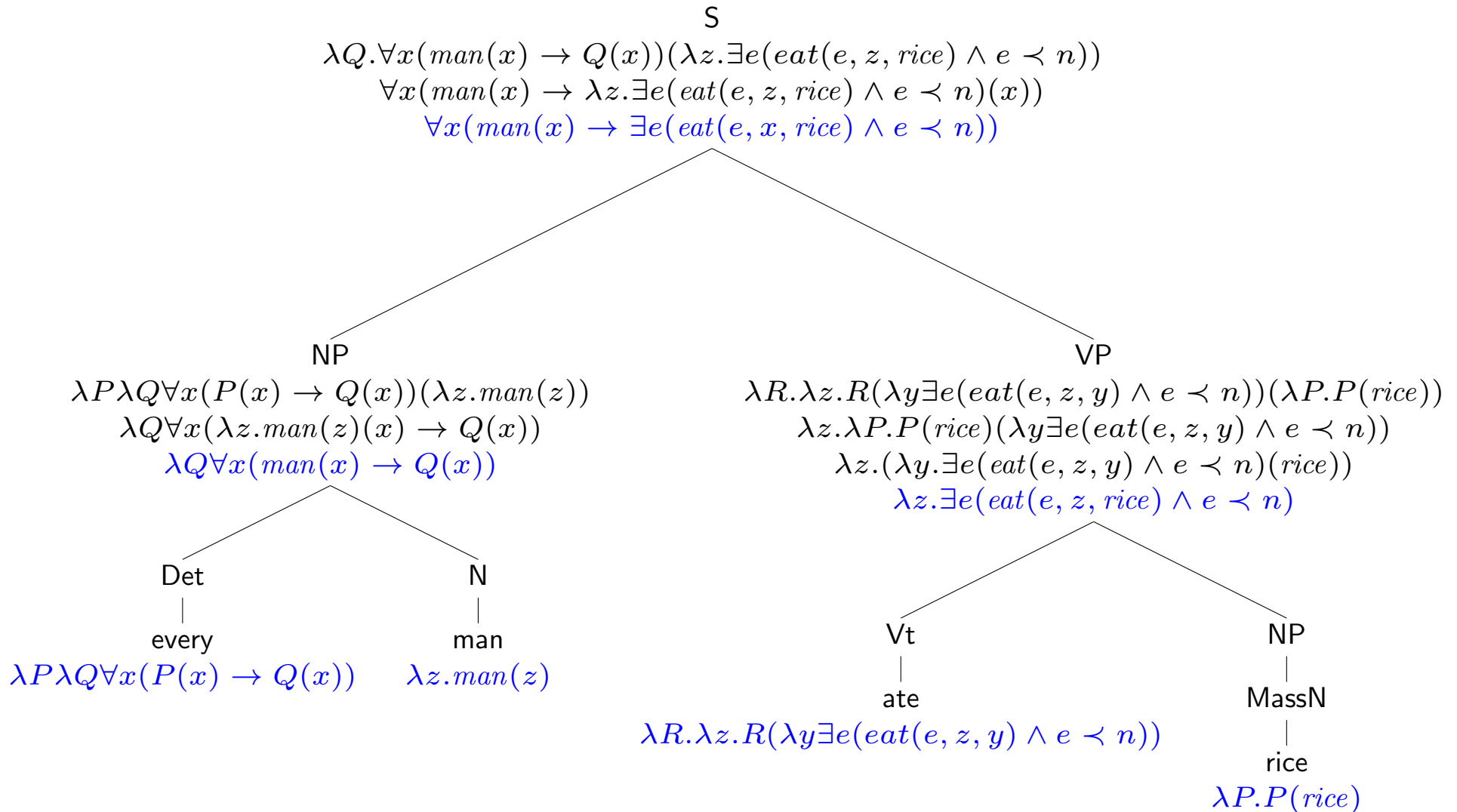
$\lambda R.\lambda z.R(\lambda y.\exists e(eat(e, z, y) \wedge e \prec n))(\lambda Q\forall x(grape(x) \rightarrow Q(x)))$ becomes
 $\lambda z\lambda Q\forall x(grape(x) \rightarrow Q(x))(\lambda y.\exists e(eat(e, z, y) \wedge e \prec n))$ becomes
 $\lambda z\forall x(grape(x) \rightarrow \lambda y.\exists e(eat(e, z, y) \wedge e \prec n)(x))$ becomes
 $\lambda z\forall x(grape(x) \rightarrow \exists e(eat(e, z, x)))$

Grammar Refined!

(Changes in purple)

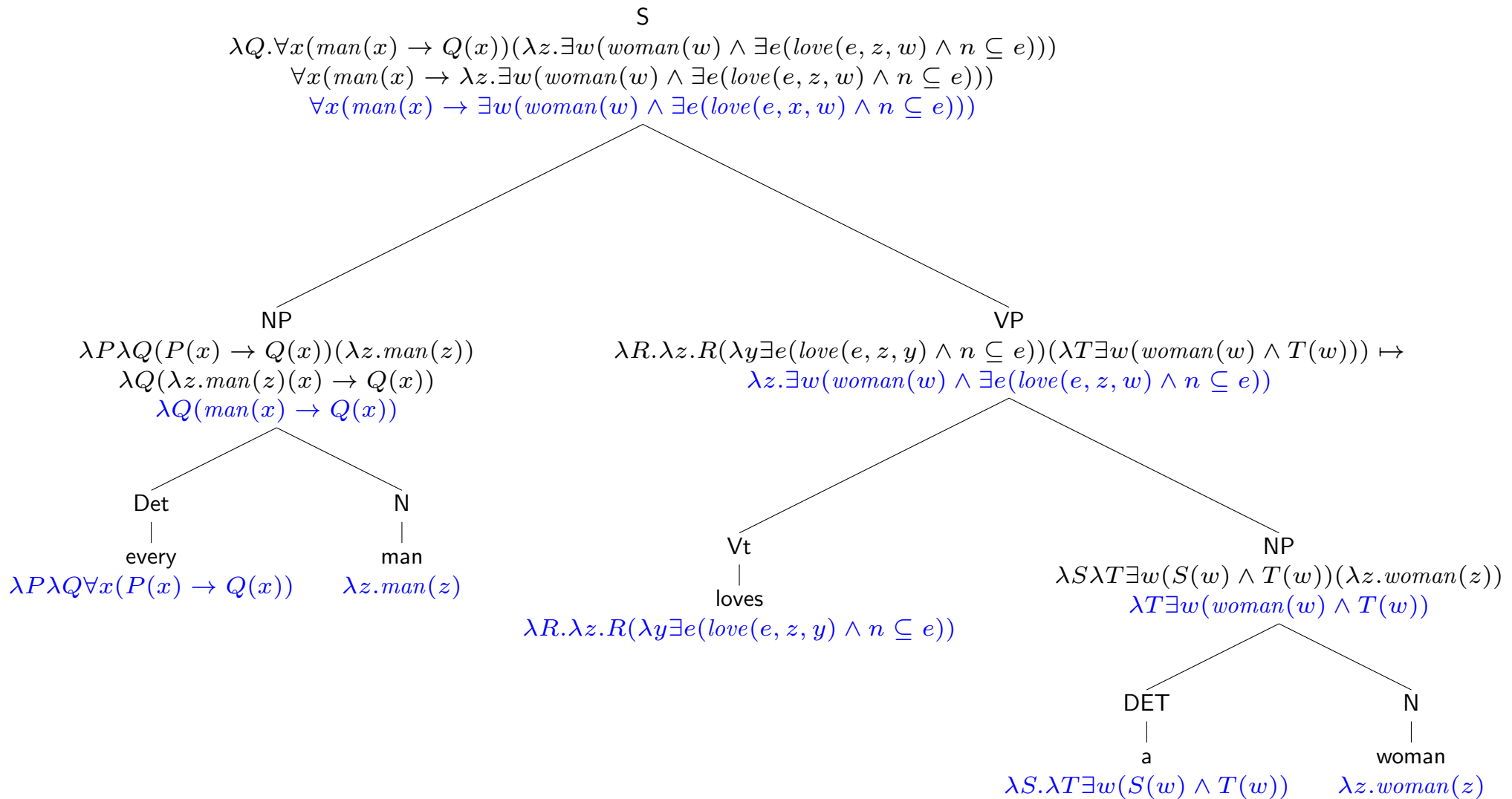
S →	NP VP <i>NP.Sem(VP.Sem)</i>	(Sentences)
NP →	MassN <i>MassN.Sem</i> PropN <i>PropN.Sem</i> <i>Det N Det.Sem(N.Sem)</i>	(Noun phrases)
VP →	Vi <i>Vi.Sem</i> Vt NP <i>Vt.Sem(NP.Sem)</i>	(Verb phrases)
PropN →	Fred <i>$\lambda P.P(\text{fred})$</i> ...	(Proper nouns)
MassN →	rice <i>$\lambda P.P(\text{rice})$</i> ...	(Mass nouns)
Vi →	talked <i>$\lambda x \exists e(\text{talk}(e, x) \wedge e \prec n)$</i> ...	(Intransitive verbs)
Vt →	ate <i>$\lambda R.\lambda z.R(\lambda y.\exists e(\text{eat}(e, z, y) \wedge e \prec n))$</i>	(Transitive verbs)
N →	man <i>$\lambda x.\text{man}(x)$</i>	(Count Nouns)
Det →	a <i>$\lambda P \lambda Q \exists x(P(x) \wedge Q(x))$</i> every <i>$\lambda Q \lambda P \exists x(P(x) \rightarrow Q(x))$</i>	(Determiners)

Example Derivation: Every man ate rice



Every man loves a woman

Other reading??



Semantic Ambiguity

- Every man loves a woman has two different interpretations because of its determiners:
 - Possibly a different woman per man
 $\forall x(\text{man}(x) \rightarrow \exists y(\text{woman}(y) \wedge \exists e(\text{love}(e, x, y) \wedge n \subseteq e)))$
 - The same woman for all men
 $\exists y(\text{woman}(y) \wedge \forall x(\text{man}(x) \rightarrow \exists e(\text{love}(e, x, y) \wedge n \subseteq e)))$
- But the English sentence isn't syntactically ambiguous!!

Scope

- The ambiguity arises because **every** and **a** each has its own **scope**:
 - Interpretation 1: **every** has scope over **a**
 - Interpretation 2: **a** has scope over **every**
- Scope is not uniquely determined either by left-to-right order, or by position in the parse tree.
- We therefore need other mechanisms to ensure that the ambiguity is reflected in the LF assigned to S.

Scope ambiguity, continued

The number of interpretations grows exponentially with the number of scope operators:

Every student at **some** university has access to **a** laptop.

1. Not necessarily same laptop, not necessarily same university

$$\forall x(stud(x) \wedge \exists y(univ(y) \wedge at(x, y)) \rightarrow \exists z(laptop(z) \wedge have_access(x, z)))$$

2. Same laptop, not necessarily same university

$$\exists z(laptop(z) \wedge \forall x(stud(x) \wedge \exists y(univ(y) \wedge at(x, y)) \rightarrow have_access(x, z)))$$

3. Not necessarily same laptop, same university

$$\exists y(univ(y) \wedge \forall x((stud(x) \wedge at(x, y)) \rightarrow \exists z(laptop(z) \wedge have_access(x, z))))$$

4. Same university, same laptop $\exists y(univ(y) \wedge \exists z(laptop(z) \wedge \forall x((stud(x) \wedge at(x, y)) \rightarrow have_access(x, z))))$

5. Same laptop, same university $\exists z(laptop(z) \wedge \exists y(univ(y) \wedge \forall x((stud(x) \wedge at(x, y)) \rightarrow have_access(x, z))))$

where 4 & 5 are equivalent

Every student at **some** university does **not** have access to **a** computer.

→ 18 interpretations

Coping with Scope: options

Enumerate all interpretations: Computationally unattractive!

Semantic Underspecification: Build LFs via syntax that **underspecify** the relative semantic scopes of the quantifiers

- Partial description of a FoL formula
- So Syntax-Tree:LF is 1:1, but the LF describes several FoL formulae and hence several interpretations

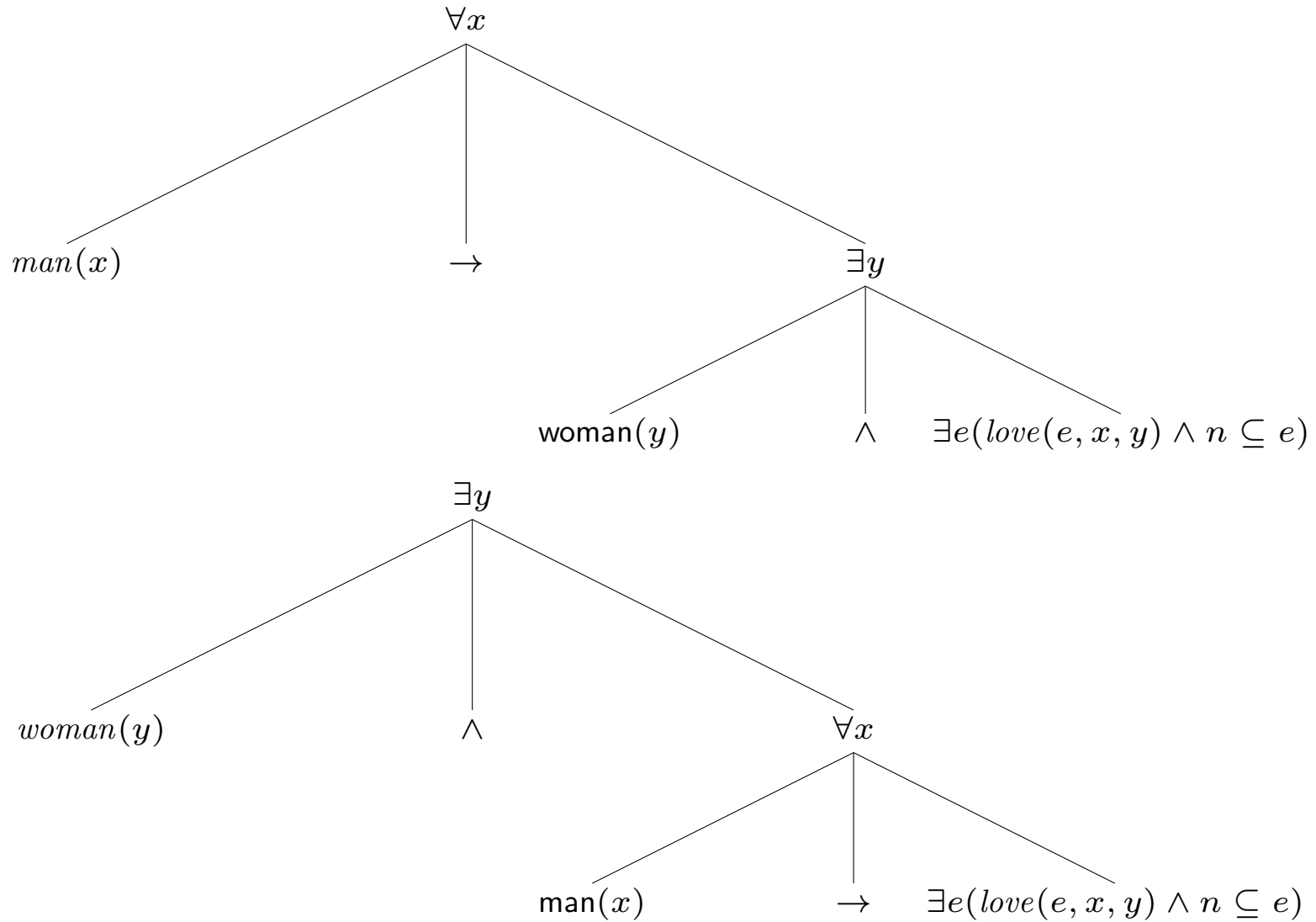
Sometimes the surrounding context will help us choose between interpretations:

Every student has access to a computer. It can be borrowed from the ITO.
(\Rightarrow a outscopes every)

Semantic Underspecification

- The LF constructed in the grammar features:
 1. FoL bits
 2. constraints on how they can combine into an FoL formula
- The constraints are satisfied by more than one FoL formula.

A Picture showing common bits and different bits



Technique

- Label nodes of the tree: $l_1, l_2 \dots$
- Supply constraints on what FoL expressions appear at those labels

Every man loves a woman.

Ignoring $\exists e$ and $n \subseteq e. \dots$

$$l_1 : \forall x(h_2 \rightarrow h_3)$$

$$l_2 : \text{man}(x)$$

$$l_3 : \text{love}(e, x, y)$$

$$l_4 : \exists y(h_4 \wedge h_5)$$

$$l_5 : \text{woman}(y)$$

$$h_2 =_q l_2, h_4 =_q l_5$$

Resolving Scope

$$\begin{aligned}l_1 &: \forall x(h_2 \rightarrow h_3) \\l_2 &: \text{man}(x) \\l_3 &: \text{love}(e, x, y) \\l_4 &: \exists y(h_4 \wedge h_5) \\l_5 &: \text{woman}(y) \\h_2 =_q l_2, h_4 =_q l_5\end{aligned}$$

- All h s must equal a (unique) l ; no free variables
- So there are two solutions:

$$\exists \text{ outscopes } \forall: h_2 = l_2, h_4 = l_5, h_3 = l_3, h_5 = l_1$$

$$\forall \text{ outscopes } \exists: h_2 = l_2, h_4 = l_5, h_3 = l_4, h_5 = l_3$$

- LF construction via the grammar must now λ -abstract labels, as well as predicates, arguments to predicates etc.

Summary

- Syntax guides semantic composition in a systematic way.
- Lambda expressions facilitate the construction of compositional semantic interpretations off the syntax tree.
 - Associate each word with a λ -term
 - Within the grammar, say which daughter is the functor.
- However, **semantic scope ambiguities** suggest that not all semantic ambiguities should surface as syntactic ambiguities within the grammar.
- There are solutions to this that exploit **semantic underspecification**.

Next Lecture: Semantic Role Labelling