Computing Natural Language Semantics

Informatics 2A: Lecture 24

Shay Cohen

17 November 2015

Compositional Semantics: the key idea

Grammar I

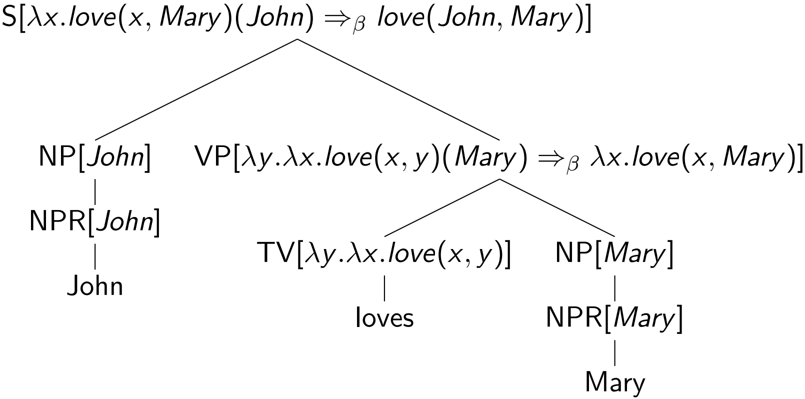
S $\rightarrow$ NP VP	{VP.Sem(NP.Sem)}	t
VP $\rightarrow$ TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
NP $\rightarrow$ NPR	{NPR.Sem}	e
TV $\rightarrow$ loves	{λy.λx.love(x,y)}	$< e, < e, t >>$
NPR $\rightarrow$ Mary	{Mary}	e
NPR $\rightarrow$ John	{John}	e

- To build a compositional semantics for NL, we attach valuation functions to grammar rules (semantic attachments).
- These show how to compute the interpretation of the LHS of the rule from the interpretations of its RHS components.
- For example, VP.Sem(NP.Sem) means apply the interpretation of the VP to the interpretation of the NP.
- Types have been added to ease understanding.

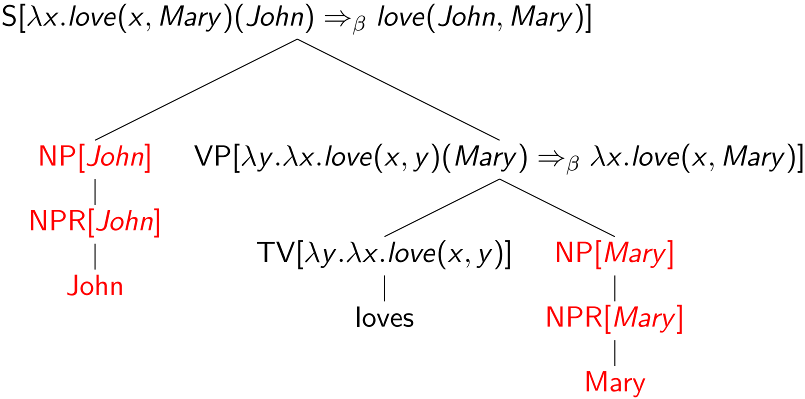
Example

The semantics of “John loves Mary”:

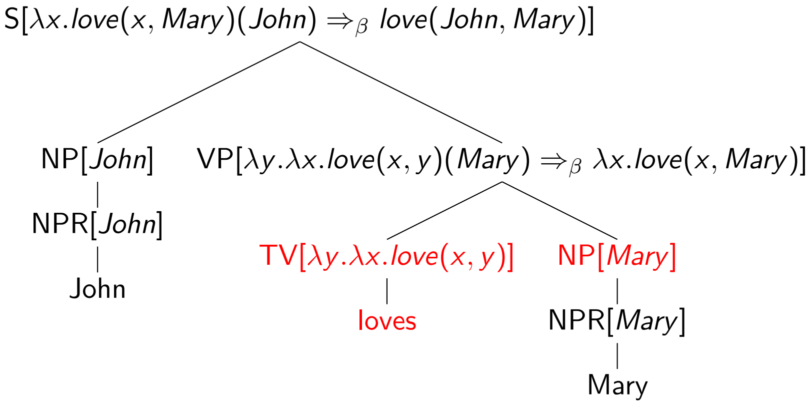
Compositional Semantics: example



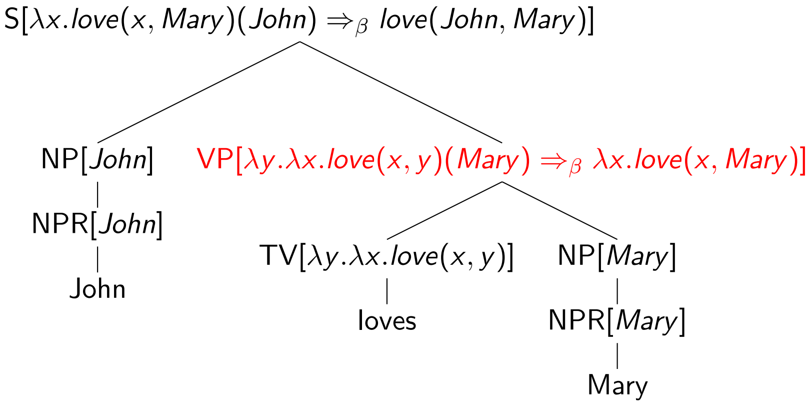
Compositional Semantics: example

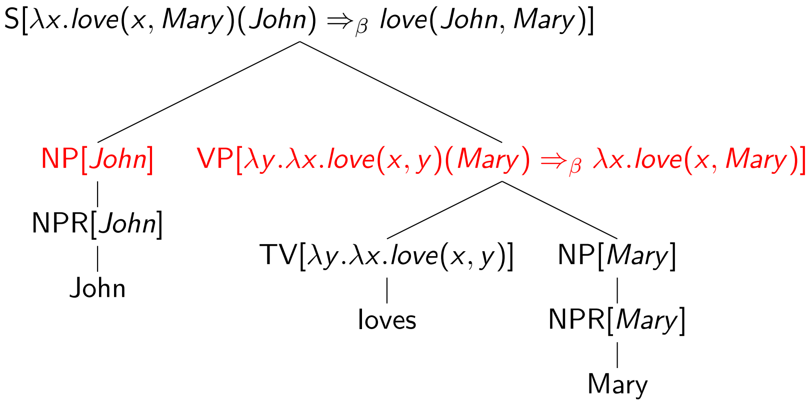


Compositional Semantics: example

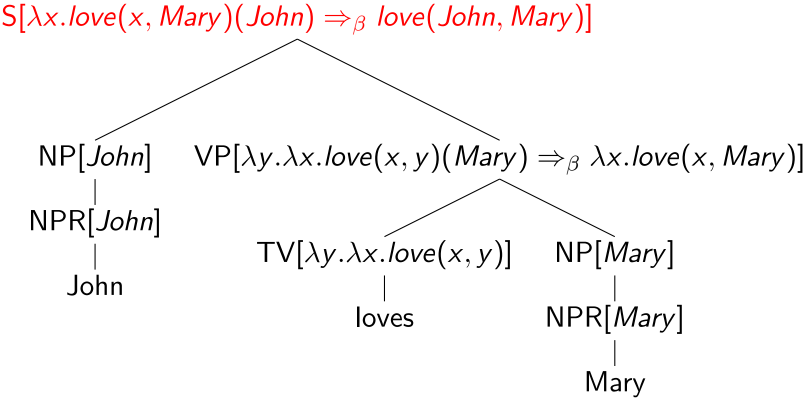


Compositional Semantics: example




S[$\lambda x.love(x, Mary)(John) \Rightarrow_\beta love(John, Mary)$]
NP[$John$]
VP[$\lambda y.\lambda x.love(x, y)(Mary) \Rightarrow_\beta \lambda x.love(x, Mary)$]
NPR[$John$]
John
TV[$\lambda y.\lambda x.love(x, y)$]
NP[$Mary$]
loves
NPR[$Mary$]
Mary

Compositional Semantics: example



A minor variation

The following alternative semantics assigns the same overall meaning to sentences. Only the treatment of the arguments of 'love' is different.

Grammar I

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{λx.TV.Sem(x)(NP.Sem)}	$< e, t >$
NP → NPR	{NPR.Sem}	e
TV → loves	{λx.λy.love(x,y)}	$< e, < e, t >>$
NPR → Mary	{Mary}	e
NPR → John	{John}	e

Compositional Semantics, continued

What about the interpretation of an NP other than a proper name? The FOPL interpretation should often contain an existential ($\exists$) or a universal ($\forall$) quantifier:

John has access to a computer.
$\exists x(computer(x) \wedge have\_access\_to(john, x))$

Every student has access to a computer.
$\forall x(student(x) \rightarrow \exists y(computer(y) \wedge have\_access\_to(x, y)))$

Can we build such interpretations up from their component parts in the same way as with proper names?

A halfway stage.

Grammar II

S $\rightarrow$ NPR VP	{ VP.Sem(NPR.Sem) }	t
VP $\rightarrow$ TV a Nom	{ $\lambda x.\exists y.$ Nom.Sem(y) & TV.Sem(y)(x) }	$< e, t >$
Nom $\rightarrow$ N	{ N.Sem }	$< e, t >$
Nom $\rightarrow$ A Nom	{ $\lambda x.$ Nom.Sem(x) & A.Sem(x) }	$< e, t >$
NPR $\rightarrow$ John	{ *John* }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ girl	{ $\lambda z.girl(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- Note we haven't given a meaning here to a tall girl.
- Could take this to have the same meaning as tall girl.
- This would be fine for this example (also in Assignment 2). But what about every tall girl?

A halfway stage.

Grammar II

S $\rightarrow$ NPR VP	{ VP.Sem(NPR.Sem) }	t
VP $\rightarrow$ TV a Nom	{ $\lambda x.\exists y.$ Nom.Sem(y) & TV.Sem(y)(x) }	$< e, t >$
Nom $\rightarrow$ N	{ N.Sem }	$< e, t >$
Nom $\rightarrow$ A Nom	{ $\lambda x.$ Nom.Sem(x) & A.Sem(x) }	$< e, t >$
NPR $\rightarrow$ John	{ *John* }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ girl	{ $\lambda z.girl(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- Note we haven't given a meaning here to a tall girl.
- Could take this to have the same meaning as tall girl.
- This would be fine for this example (also in Assignment 2). But what about every tall girl?

A halfway stage.

Grammar II

S $\rightarrow$ NPR VP	{ VP.Sem(NPR.Sem) }	t
VP $\rightarrow$ TV a Nom	{ $\lambda x.\exists y.$ Nom.Sem(y) & TV.Sem(y)(x) }	$< e, t >$
Nom $\rightarrow$ N	{ N.Sem }	$< e, t >$
Nom $\rightarrow$ A Nom	{ $\lambda x.$ Nom.Sem(x) & A.Sem(x) }	$< e, t >$
NPR $\rightarrow$ John	{ *John* }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ girl	{ $\lambda z.girl(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- Note we haven't given a meaning here to a tall girl.
- Could take this to have the same meaning as tall girl.
- This would be fine for this example (also in Assignment 2). But what about every tall girl?

A halfway stage.

Grammar II

S $\rightarrow$ NPR VP	{ VP.Sem(NPR.Sem) }	t
VP $\rightarrow$ TV a Nom	{ $\lambda x.\exists y.$ Nom.Sem(y) & TV.Sem(y)(x) }	$< e, t >$
Nom $\rightarrow$ N	{ N.Sem }	$< e, t >$
Nom $\rightarrow$ A Nom	{ $\lambda x.$ Nom.Sem(x) & A.Sem(x) }	$< e, t >$
NPR $\rightarrow$ John	{ $John$ }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ girl	{ $\lambda z.girl(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- Note we haven't given a meaning here to a tall girl.
- Could take this to have the same meaning as tall girl.
- This would be fine for this example (also in Assignment 2). But what about every tall girl?

The semantics of “John loves a tall girl”:

Computing semantics with Grammar II

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV		$\lambda yx.\ love(x,y)$
tall girl	Nom		$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
		$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP		$\lambda x.\exists y.\ (\lambda x.\ girl(x)\,\&\,tall(x))(y)\ \&$ $(\lambda yx.\ love(x,y))(y)(x)$
		$\Rightarrow_\beta$	$\lambda x.\exists y.\ (girl(y)\,\&\,tall(y))\ \&$ $love(x,y)$
John loves a tall girl	S		$(\lambda x.\exists y.\ \cdots)(John)$
		$\Rightarrow_\beta$	$\exists y.girl(y)\,\&\,tall(y)\,\&\,love(John,y)$

Computing semantics with Grammar II

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV		$\lambda yx.\ love(x, y)$
tall girl	Nom		$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
		$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP		$\lambda x.\exists y.\ (\lambda x.\ girl(x)\ \&\ tall(x))(y)\ \&$ $(\lambda yx.\ love(x, y))(y)(x)$
		$\Rightarrow_\beta$	$\lambda x.\exists y.\ (girl(y)\ \&\ tall(y))\ \&$ $love(x, y)$
John loves a tall girl	S		$(\lambda x.\exists y.\ \cdots)(John)$
		$\Rightarrow_\beta$	$\exists y.girl(y)\ \&\ tall(y)\ \&\ love(John, y)$

Computing semantics with Grammar II

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV	$\lambda yx.\ love(x,y)$
tall girl	Nom	$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
	$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP	$\lambda x.\exists y.\ (\lambda x.\ girl(x)\ \&\ tall(x))(y)\ \&$ $(\lambda yx.\ love(x,y))(y)(x)$
	$\Rightarrow_\beta$	$\lambda x.\exists y.\ (girl(y)\ \&\ tall(y))\ \&$ $love(x,y)$
John loves a tall girl	S	$(\lambda x.\exists y.\ \cdots)(John)$
	$\Rightarrow_\beta$	$\exists y.girl(y)\ \&\ tall(y)\ \&\ love(John,y)$

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV	$\lambda yx.\ love(x, y)$
tall girl	Nom	$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
	$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP	$\lambda x.\exists y.\ (\lambda x.\ girl(x)\ \&\ tall(x))(y)\ \&$ $(\lambda yx.\ love(x, y))(y)(x)$
	$\Rightarrow_\beta$	$\lambda x.\exists y.\ (girl(y)\ \&\ tall(y))\ \&$ $love(x, y)$
John loves a tall girl	S	$(\lambda x.\exists y.\ \cdots)(John)$
	$\Rightarrow_\beta$	$\exists y.girl(y)\ \&\ tall(y)\ \&\ love(John, y)$

Computing semantics with Grammar II

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV	$\lambda yx.\ love(x, y)$
tall girl	Nom	$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
	$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP	$\lambda x.\exists y.\ (\lambda x.\ girl(x)\ \&\ tall(x))(y)\ \&$ $(\lambda yx.\ love(x, y))(y)(x)$
	$\Rightarrow_\beta$	$\lambda x.\exists y.\ (girl(y)\ \&\ tall(y))\ \&$ $love(x, y)$
John loves a tall girl	S	$(\lambda x.\exists y.\ \cdots)(John)$
	$\Rightarrow_\beta$	$\exists y.girl(y)\ \&\ tall(y)\ \&\ love(John, y)$

Computing semantics with Grammar II

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV		$\lambda yx.\ love(x, y)$
tall girl	Nom		$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
		$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP		$\lambda x. \exists y.\ (\lambda x.\ girl(x)\ \&\ tall(x))(y)\ \&$ $(\lambda yx.\ love(x, y))(y)(x)$
		$\Rightarrow_\beta$	$\lambda x. \exists y.\ (girl(y)\ \&\ tall(y))\ \&$ $love(x, y)$
John loves a tall girl	S		$(\lambda x. \exists y.\ \cdots)(John)$
		$\Rightarrow_\beta$	$\exists y.girl(y)\ \&\ tall(y)\ \&\ love(John, y)$

Computing semantics with Grammar II

Before we add more, let's use Grammar II to compute the semantics of John loves a tall girl.

loves	TV		$\lambda yx.\ love(x, y)$
tall girl	Nom		$\lambda x.\ (\lambda z.girl(z))(x)\ \&\ (\lambda z.tall(z))(x)$
		$\Rightarrow_\beta$	$\lambda x.\ girl(x)\ \&\ tall(x)$
loves a tall girl	VP		$\lambda x.\exists y.\ (\lambda x.\ girl(x)\ \&\ tall(x))(y)\ \&$ $(\lambda yx.\ love(x, y))(y)(x)$
		$\Rightarrow_\beta$	$\lambda x.\exists y.\ (girl(y)\ \&\ tall(y))\ \&$ $love(x, y)$
John loves a tall girl	S		$(\lambda x.\exists y.\ \cdots)(John)$
		$\Rightarrow_\beta$	$\exists y.girl(y)\ \&\ tall(y)\ \&\ love(John, y)$

Type raising

- We've given John, Mary the semantic type e, and girl the semantic type $< e, t >$.
- But what type should some girl or every girl have?
- Idea: Since we wish to combine an NP.Sem with a VP.Sem (of type $< e, t >$) to get an S.Sem (of type t), let's try again with NP.Sem having type $<< e, t >, t >$.

John	$\lambda P.\, P(John)$	(type raising)
every girl	$\lambda P.\, \forall x.\, girl(x) \Rightarrow P(x)$	

The appropriate semantic attachment for NP VP is then

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

Semantics of determiners

- Using this approach, we can also derive the semantics of 'every girl' from that of 'every' and 'girl'.
- We've seen that 'girl' has semantic type $< e, t >$, and 'every girl' has semantic type $<< e, t >, t >$.
- So the interpretation of 'every' should have type $<< e, t >, << e, t >, t >>$. Similarly for other *determiners* (*e.g.* every, a, no, not every).

girl	$\lambda x.$ girl(x)	$< e, t >$
every	$\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x)$	$<< e, t >, << e, t >, t >>$
a	$\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x)$	$<< e, t >, << e, t >, t >>$
NP $\rightarrow$ Det N	{ Det.Sem (N.Sem) }	$<< e, t >, t >$

We can now compute the semantics of 'every girl' and check that it β-reduces to $\lambda P.\forall x.\, girl(x) \Rightarrow P(x)$.

Example

The semantics of "every girl":

More on type raising

- The natural rule for VP is now VP $\rightarrow$ TV NP.
- Since the semantic type for NP has now been raised to $<< e, t >, t >$, and we want VP to have semantic type $< e, t >$, what should the semantic type for TV be?

More on type raising

- The natural rule for VP is now VP $\rightarrow$ TV NP.
- Since the semantic type for NP has now been raised to $<< e, t >, t >$, and we want VP to have semantic type $< e, t >$, what should the semantic type for TV be?

It had better be $<<< e, t >, t >, < e, t >>$.
(A 3rd order function type!)

$$\text{TV} \rightarrow \text{loves} \quad \{\lambda R^{<<e,t>,t>}.\lambda z^e.\, R(\lambda w^e.\, loves(z, w))\}$$
$$\text{VP} \rightarrow \text{TV NP} \quad \{\text{TV.Sem(NP.Sem)}\}$$

To summarize where we've got to:

Grammar III

S $\rightarrow$ NP VP	{ NP.Sem(VP.Sem) }	t
VP $\rightarrow$ TV NP	{ TV.Sem(NP.Sem) }	$< e, t >$
NP $\rightarrow$ John	{ $\lambda P.P(John)$ }	$<< e, t >, t >$
NP $\rightarrow$ Det Nom	{ Det.Sem(Nom.Sem) }	$<< e, t >, t >$
Det $\rightarrow$ a	{ $\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x)$ }	$<< e, t >, << e, t >, t >>>$
Det $\rightarrow$ every	{ $\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x)$ }	$<< e, t >, << e, t >, t >>>$
Nom $\rightarrow$ N	{ N.Sem }	$< e, t >$
Nom $\rightarrow$ A Nom	{ $\lambda x.$ Nom.Sem(x) & A.Sem(x) }	$< e, t >$
TV $\rightarrow$ loves	{ $\{\lambda R.\lambda z.\ R(\lambda w.\ loves(z, w))\}$	$<<< e, t >, t >, < e, t >>$
N $\rightarrow$ girl	{ $\lambda z.girl(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

Can add similar entries for 'student', 'computer', 'has access to'.

Example

The semantics for ‘every student has access to a computer’.

Example

The semantics for ‘every student has access to a computer’.

every student		$(\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x))(\lambda x.\mathit{student}(x))$
	$\rightarrow_\beta$	$\lambda P.\forall x.\,\mathit{student}(x) \Rightarrow P(x)$

Example

The semantics for ‘every student has access to a computer’.

every student $(\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x))(\lambda x.student(x))$

$\rightarrow_\beta \quad \lambda P.\forall x.\, student(x) \Rightarrow P(x)$

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$

$\rightarrow_\beta \quad \lambda P.\exists x.\, computer(x) \wedge P(x)$

Example

The semantics for ‘every student has access to a computer’.

every student $(\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x))(\lambda x.student(x))$
$\rightarrow_\beta \quad \lambda P.\forall x.\, student(x) \Rightarrow P(x)$

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$
$\rightarrow_\beta \quad \lambda P.\exists x.\, computer(x) \wedge P(x)$

h.a.t. a computer $\cdots \rightarrow_\beta \cdots$
$\rightarrow_\beta \lambda z.\exists x.\, computer(x) \wedge h\_a\_t(z, x)$

Example

The semantics for 'every student has access to a computer'.

every student $(\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x))(\lambda x.student(x))$
$\rightarrow_\beta \quad \lambda P.\forall x.\, student(x) \Rightarrow P(x)$

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$
$\rightarrow_\beta \quad \lambda P.\exists x.\, computer(x) \wedge P(x)$

h.a.t. a computer $\cdots \rightarrow_\beta \cdots$
$\rightarrow_\beta \lambda z.\exists x.\, computer(x) \wedge h\_a\_t(z, x)$

(whole sentence) $\cdots \rightarrow_\beta \cdots$
$\rightarrow_\beta \forall x.\, student(x) \Rightarrow \exists y.\, computer(y) \wedge h\_a\_t(x, y)$

Example

The semantics for 'every student has access to a computer'.

every student $(\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x))(\lambda x.student(x))$

$\rightarrow_\beta \quad \lambda P.\forall x.\, student(x) \Rightarrow P(x)$

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$

$\rightarrow_\beta \quad \lambda P.\exists x.\, computer(x) \wedge P(x)$

h.a.t. a computer $\cdots \rightarrow_\beta \cdots$

$\rightarrow_\beta \lambda z.\exists x.\, computer(x) \wedge h\_a\_t(z, x)$

(whole sentence) $\cdots \rightarrow_\beta \cdots$

$\rightarrow_\beta \forall x.\, student(x) \Rightarrow \exists y.\, computer(y) \wedge h\_a\_t(x, y)$

Note: In the last β-step, we've renamed 'x' to 'y' to avoid capture.

Question

Suppose that the predicate L(x, y) means x loves y. Which of the following is not a possible representation of the meaning of *Everybody loves somebody*?

1. $\forall x.\exists y.L(x, y)$
2. $(\lambda P.\forall x.\exists y.P(x, y))(\lambda x.\lambda y.L(x, y))$
3. $(\lambda P.\forall x.\exists y.P(x, y))(\lambda x.\lambda y.L(y, x))$
4. $(\lambda P.\forall x.\exists y.P(y, x))(\lambda x.\lambda y.L(y, x))$

Semantic Ambiguity

Whilst every student has access to a computer is neither syntactically nor lexically ambiguous, it has two different interpretations because of its determiners:

- every: interpreted as $\forall$ *(universal quantifier)*
- a: interpreted as $\exists$ *(existential quantifier)*

Meaning 1

Possibly a different computer per student

$\forall x(student(x) \rightarrow \exists y(computer(y) \wedge have\_access\_to(x, y)))$

Meaning 2

Possibly the same computer for all students

$\exists y(computer(y) \wedge \forall x(student(y) \rightarrow have\_access\_to(x, y)))$

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 by there being multiple interpretations 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
$\forall x(\mathit{stud}(x) \wedge \exists y(\mathit{univ}(y) \wedge \mathit{at}(x,y)) \rightarrow \exists z(\mathit{laptop}(z) \wedge \mathit{have\_access}(x,z)))$
2. Same laptop, not necessarily same university
$\exists z(\mathit{laptop}(z) \wedge \forall x(\mathit{stud}(x) \wedge \exists y(\mathit{univ}(y) \wedge \mathit{at}(x,y)) \rightarrow \mathit{have\_access}(x,z)))$
3. Not necessarily same laptop, same university
$\exists y(\mathit{univ}(y) \wedge \forall x((\mathit{stud}(x) \wedge \mathit{at}(x,y)) \rightarrow \exists z(\mathit{laptop}(z) \wedge \mathit{have\_access}(x,z))))$
4. Same university, same laptop
$\exists y(\mathit{univ}(y) \wedge \exists z(\mathit{laptop}(z) \wedge \forall x((\mathit{stud}(x) \wedge \mathit{at}(x,y)) \rightarrow \mathit{have\_access}(x,z))))$
5. Same laptop, same university
$\exists z(\mathit{laptop}(z) \wedge \exists y(\mathit{univ}(y) \wedge \forall x((\mathit{stud}(x) \wedge \mathit{at}(x,y)) \rightarrow \mathit{have\_access}(x,z))))$

where 4 & 5 are equivalent

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

$\rightarrow$ 18 interpretations

Coping with Scope: options

1. **Enumerate all interpretations**. Computationally unattractive!
2. Use an **underspecified representation** that can be further specified to each of the multiple interpretations on demand.

Sometimes the surrounding context will help us choose between interpretations:

> Every student has access to a computer. It can be borrowed from the ITO. ($\Rightarrow$ Meaning 2)

Underspecification

- The idea in underspecified representations is that instead of trying to associate a single FOPL formula with a sentence, we associate fragments of formulae with various parts of the sentence.
- These fragments can have holes into which other fragments can be plugged. Since there may be some freedom in the order of plugging, the same bunch of fragments can give rise to several formulae with different scoping orders.
- There may also be constraints on the order of plugging, corresponding to partial information about the intended interpretation derived e.g. from the discourse context.

See J&M Chapter 18.3 for more on this.

Summary

- Syntax guides semantic composition in a systematic way.
- Lambda expressions facilitate the construction of compositional semantic interpretations.
- Logical forms can be constructed by attaching valuation functions to grammar rules.
- However, this approach is not adequate enough for quantified NPs, as LFs are not always isomorphic with syntax.
- We can elegantly handle scope by building an abstract underspecified representation and disambiguate on demand.