

Computing Natural Language Semantics

Informatics 2A: Lecture 26

Shay Cohen

14 November 2017

1 Semantic Composition

- Review: compositionality, lambda expressions, and logical forms
- Examples
- Type raising

2 Semantic (Scope) Ambiguity

- Definition
- Semantic Scope
- Approaches to Scope Ambiguity

- Consider the question “Who is the CEO of Microsoft?”
- A possible semantic interpretation of it:
 $\lambda x. \text{CEO}(x, \text{Microsoft})$
- In addition, we have a large database of CEOs, in relations of the form $\text{CEO}(\text{Nadella}, \text{Microsoft})$, $\text{CEO}(\text{Cook}, \text{Apple})$, etc.
- Given the above λ expression, we can apply it on all entities in the database and check its truth value (i.e. whether in our *model* of the world, as reflected by the database, the CEO relation holds between the entity and Microsoft).

How do we create FOL for sentences in a compositional manner?
Through their syntactic tree.

Each node in the tree will have a lambda expression associated with it (or a logical form). That expression has a “type” – it is sometimes easier to think in terms of “what type a node should be,” instead of “what formula it should have.”

Roadmap: (1) simple case of statements such as “John loves Mary.” (2) how do we tackle sentences such as “John loves a tall woman?” (3) how do we tackle sentences such as “Every man loves a woman?”

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.

Do we have sufficient tools to systematically compute meaning representations according to this principle?

- The meaning of a complete sentence will hopefully be a FOPL formula, which we consider as having type t (truth values).
- But the meaning of smaller fragments of the sentence will have other types. E.g.

has a bone $\langle e, t \rangle$

every dog $\langle \langle e, t \rangle, t \rangle$

- The idea is to show how to associate a meaning with such fragments, and how these meanings combine.
- To do this, we need to extend our language of FOPL with λ expressions ($\lambda =$ lambda; written as `\` in Haskell).

Lambda (λ) Expressions

λ -**expressions** are an extension to FOPL that allows us to work with '**partially constructed**' formulae. A λ -expression consists of:

- the Greek letter λ , followed by a variable (**formal parameter**);
- a FOPL expression that may involve that variable.

$\lambda x.sleep(x) : \langle e, t \rangle$

'The function that takes an entity x to the statement $sleep(x)$ '

$\underbrace{(\lambda x.sleep(x))}_{function} \underbrace{(Kim)}_{argument} : t$

A λ -expression can be **applied** to a term.

(The above has the same truth value as $sleep(Kim)$.)

Lambda expressions can be **nested**. We can use nesting to create functions of several arguments that accept their arguments one at a time.

$\lambda y. \lambda x. \text{love}(x,y) : \langle e, \langle e, t \rangle \rangle$

'The function that takes y to
(the function that takes x to the statement $\text{love}(x,y)$)'

$\lambda z. \lambda y. \lambda x. \text{give}(x,y,z) : \langle e, \langle e, \langle e, t \rangle \rangle \rangle$

'The function that takes z to
(the function that takes y to
(the function that takes x to the statement $\text{give}(x,y,z)$))'

Beta Reduction

When a lambda expression **applies** to a term, a reduction operation (**beta (β) reduction**) can be used to replace its formal parameter with the term and simplify the result. In general:

$$(\lambda x.M)N \Rightarrow_{\beta} M[x \mapsto N] \quad (M \text{ with } N \text{ substituted for } x)$$

$$\underbrace{(\lambda x.sleep(x))}_{\text{function}} \underbrace{(Kim)}_{\text{argument}} \Rightarrow_{\beta} sleep(Kim)$$

$$\underbrace{(\lambda y.\lambda x.love(x, y))}_{\text{function}} \underbrace{(crabapples)}_{\text{argument}} \Rightarrow_{\beta} \lambda x.love(x, crabapples)$$

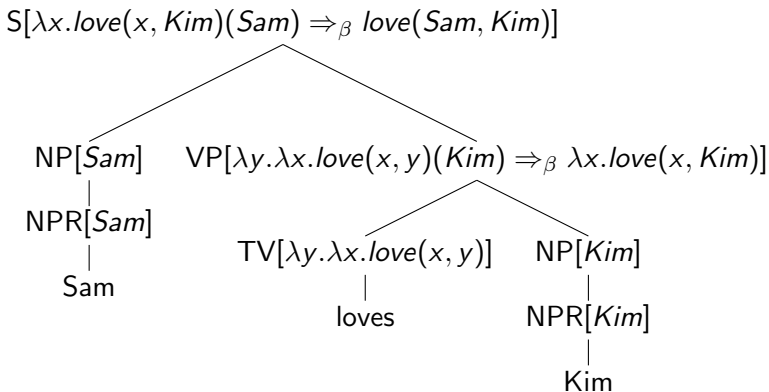
$$\underbrace{(\lambda x.love(x, crabapples))}_{\text{function}} \underbrace{(Kim)}_{\text{argument}} \Rightarrow_{\beta} love(Kim, crabapples)$$

Grammar I

$S \rightarrow NP VP$	$\{VP.Sem(NP.Sem)\}$	t
$VP \rightarrow TV NP$	$\{TV.Sem(NP.Sem)\}$	$\langle e, t \rangle$
$NP \rightarrow NPR$	$\{NPR.Sem\}$	e
$TV \rightarrow \text{loves}$	$\{\lambda y. \lambda x. \text{love}(x,y)\}$	$\langle e, \langle e, t \rangle \rangle$
$NPR \rightarrow \text{Kim}$	$\{\text{Kim}\}$	e
$NPR \rightarrow \text{Sam}$	$\{\text{Sam}\}$	e

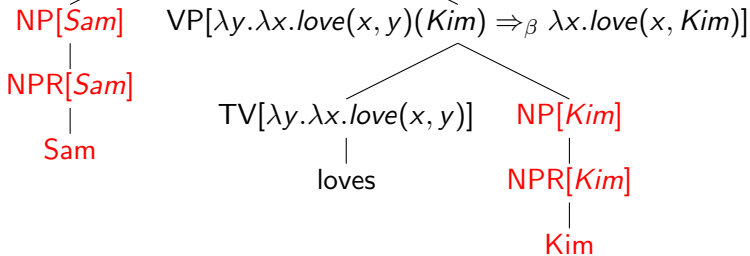
- To build a compositional semantics for NL, we attach **valuation functions** to grammar rules (**semantic attachments**).
- These show how to compute the interpretation of the LHS of the rule from the interpretations of its RHS components.
- For example, $VP.Sem(NP.Sem)$ means **apply** the interpretation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

Compositional Semantics: example



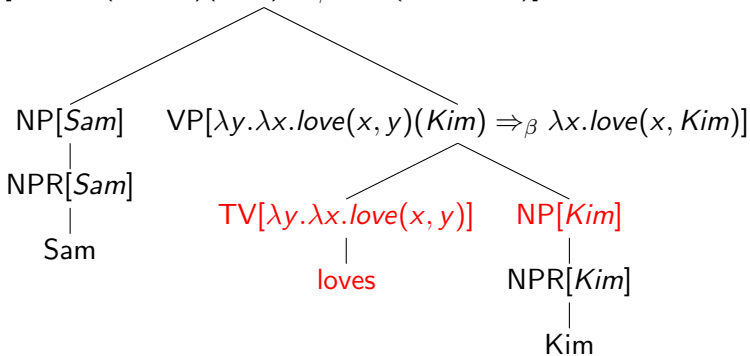
Compositional Semantics: example

$S[\lambda x.love(x, Kim)(Sam) \Rightarrow_{\beta} love(Sam, Kim)]$



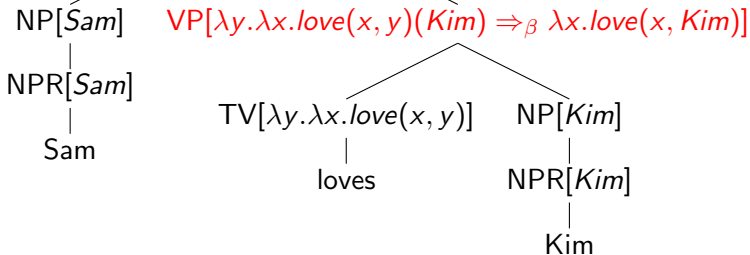
Compositional Semantics: example

$S[\lambda x.love(x, Kim)(Sam) \Rightarrow_{\beta} love(Sam, Kim)]$



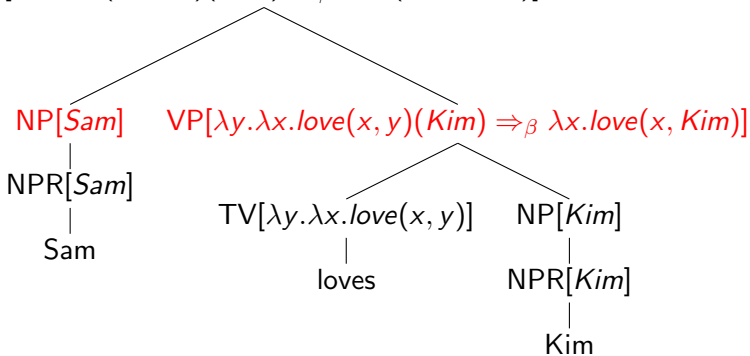
Compositional Semantics: example

$S[\lambda x.love(x, Kim)(Sam) \Rightarrow_{\beta} love(Sam, Kim)]$

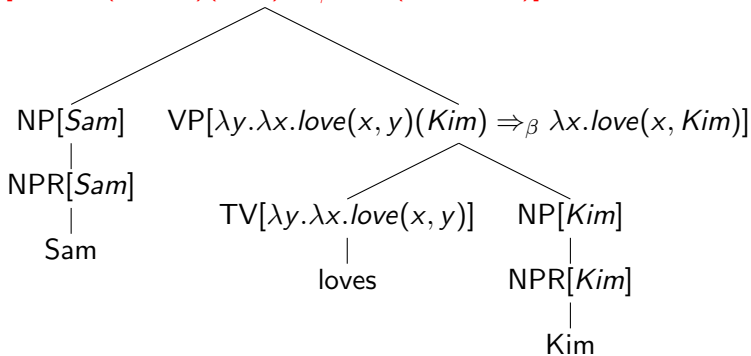


Compositional Semantics: example

$S[\lambda x.love(x, Kim)(Sam) \Rightarrow_{\beta} love(Sam, Kim)]$



$S[\lambda x.love(x, Kim)(Sam) \Rightarrow_{\beta} love(Sam, Kim)]$



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

$S \rightarrow NP VP$	$\{VP.Sem(NP.Sem)\}$	t
$VP \rightarrow TV NP$	$\{\lambda x.TV.Sem(x)(NP.Sem)\}$	$\langle e, t \rangle$
$NP \rightarrow NPR$	$\{NPR.Sem\}$	e
$TV \rightarrow loves$	$\{\lambda x.\lambda y.love(x,y)\}$	$\langle e, \langle e, t \rangle \rangle$
$NPR \rightarrow Kim$	$\{Kim\}$	e
$NPR \rightarrow Sam$	$\{Sam\}$	e

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:

Sam has access to **a computer**.

$\exists x(\text{computer}(x) \wedge \text{have_access_to}(\text{Sam}, x))$

Every student has access to a computer.

$\forall x(\text{student}(x) \rightarrow \exists y(\text{computer}(y) \wedge \text{have_access_to}(x, y)))$

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

A halfway stage.

Grammar II

$S \rightarrow \text{NPR VP}$	$\{ \text{VP.Sem}(\text{NPR.Sem}) \}$	t
$\text{VP} \rightarrow \text{TV a Nom}$	$\{ \lambda x. \exists y. \text{Nom.Sem}(y) \&$ $\text{TV.Sem}(y)(x) \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{N}$	$\{ \text{N.Sem} \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{A Nom}$	$\{ \lambda x. \text{Nom.Sem}(x) \& \text{A.Sem}(x) \}$	$\langle e, t \rangle$
$\text{NPR} \rightarrow \text{Sam}$	$\{ \text{Sam} \}$	e
$\text{TV} \rightarrow \text{loves}$	$\{ \lambda y. \lambda x. \text{love}(x, y) \}$	$\langle e, \langle e, t \rangle \rangle$
$\text{N} \rightarrow \text{woman}$	$\{ \lambda z. \text{woman}(z) \}$	$\langle e, t \rangle$
$\text{A} \rightarrow \text{tall}$	$\{ \lambda z. \text{tall}(z) \}$	$\langle e, t \rangle$

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

A halfway stage.

Grammar II

$S \rightarrow \text{NPR VP}$	$\{ \text{VP.Sem}(\text{NPR.Sem}) \}$	t
$\text{VP} \rightarrow \text{TV a Nom}$	$\{ \lambda x. \exists y. \text{Nom.Sem}(y) \&$ $\text{TV.Sem}(y)(x) \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{N}$	$\{ \text{N.Sem} \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{A Nom}$	$\{ \lambda x. \text{Nom.Sem}(x) \& \text{A.Sem}(x) \}$	$\langle e, t \rangle$
$\text{NPR} \rightarrow \text{Sam}$	$\{ \text{Sam} \}$	e
$\text{TV} \rightarrow \text{loves}$	$\{ \lambda y. \lambda x. \text{love}(x, y) \}$	$\langle e, \langle e, t \rangle \rangle$
$\text{N} \rightarrow \text{woman}$	$\{ \lambda z. \text{woman}(z) \}$	$\langle e, t \rangle$
$\text{A} \rightarrow \text{tall}$	$\{ \lambda z. \text{tall}(z) \}$	$\langle e, t \rangle$

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

A halfway stage.

Grammar II

$S \rightarrow \text{NPR VP}$	$\{ \text{VP.Sem}(\text{NPR.Sem}) \}$	t
$\text{VP} \rightarrow \text{TV a Nom}$	$\{ \lambda x. \exists y. \text{Nom.Sem}(y) \&$ $\text{TV.Sem}(y)(x) \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{N}$	$\{ \text{N.Sem} \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{A Nom}$	$\{ \lambda x. \text{Nom.Sem}(x) \& \text{A.Sem}(x) \}$	$\langle e, t \rangle$
$\text{NPR} \rightarrow \text{Sam}$	$\{ \text{Sam} \}$	e
$\text{TV} \rightarrow \text{loves}$	$\{ \lambda y. \lambda x. \text{love}(x, y) \}$	$\langle e, \langle e, t \rangle \rangle$
$\text{N} \rightarrow \text{woman}$	$\{ \lambda z. \text{woman}(z) \}$	$\langle e, t \rangle$
$\text{A} \rightarrow \text{tall}$	$\{ \lambda z. \text{tall}(z) \}$	$\langle e, t \rangle$

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

A halfway stage.

Grammar II

$S \rightarrow \text{NPR VP}$	$\{ \text{VP.Sem}(\text{NPR.Sem}) \}$	t
$\text{VP} \rightarrow \text{TV a Nom}$	$\{ \lambda x. \exists y. \text{Nom.Sem}(y) \& \text{TV.Sem}(y)(x) \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{N}$	$\{ \text{N.Sem} \}$	$\langle e, t \rangle$
$\text{Nom} \rightarrow \text{A Nom}$	$\{ \lambda x. \text{Nom.Sem}(x) \& \text{A.Sem}(x) \}$	$\langle e, t \rangle$
$\text{NPR} \rightarrow \text{Sam}$	$\{ \text{Sam} \}$	e
$\text{TV} \rightarrow \text{loves}$	$\{ \lambda y. \lambda x. \text{love}(x, y) \}$	$\langle e, \langle e, t \rangle \rangle$
$\text{N} \rightarrow \text{woman}$	$\{ \lambda z. \text{woman}(z) \}$	$\langle e, t \rangle$
$\text{A} \rightarrow \text{tall}$	$\{ \lambda z. \text{tall}(z) \}$	$\langle e, t \rangle$

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

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$ $\Rightarrow_{\beta} \lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$ $\Rightarrow_{\beta} \lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$ $\Rightarrow_{\beta} \exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

Computing semantics with Grammar II

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$ $\Rightarrow_{\beta} \lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$ $\Rightarrow_{\beta} \lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$ $\Rightarrow_{\beta} \exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

Computing semantics with Grammar II

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$
	\Rightarrow_{β}	$\lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$
	\Rightarrow_{β}	$\lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$
	\Rightarrow_{β}	$\exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$
	\Rightarrow_{β}	$\lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$
	\Rightarrow_{β}	$\lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$
	\Rightarrow_{β}	$\exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

Computing semantics with Grammar II

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$
	\Rightarrow_{β}	$\lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$
	\Rightarrow_{β}	$\lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$
	\Rightarrow_{β}	$\exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

Computing semantics with Grammar II

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$ $\Rightarrow_{\beta} \lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$ $\Rightarrow_{\beta} \lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$ $\Rightarrow_{\beta} \exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

Computing semantics with Grammar II

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

loves	TV	$\lambda yx. \text{love}(x, y)$
tall woman	Nom	$\lambda x. (\lambda z. \text{woman}(z))(x) \ \& \ (\lambda z. \text{tall}(z))(x)$ $\Rightarrow_{\beta} \lambda x. \text{woman}(x) \ \& \ \text{tall}(x)$
loves a tall woman	VP	$\lambda x. \exists y. (\lambda x. \text{woman}(x) \ \& \ \text{tall}(x))(y) \ \& \ (\lambda yx. \text{love}(x, y))(y)(x)$ $\Rightarrow_{\beta} \lambda x. \exists y. (\text{woman}(y) \ \& \ \text{tall}(y)) \ \& \ \text{love}(x, y)$
Sam loves a tall woman	S	$(\lambda x. \exists y. \dots)(\text{Sam})$ $\Rightarrow_{\beta} \exists y. \text{woman}(y) \ \& \ \text{tall}(y) \ \& \ \text{love}(\text{Sam}, y)$

Type raising

- We've given **Sam**, **Kim** the semantic type e , and **woman** the semantic type $\langle e, t \rangle$.
- But what type should **some woman** or **every woman** have?
- **Idea:** Since we wish to combine an NP.Sem with a VP.Sem (of type $\langle e, t \rangle$) to get an S.Sem (of type t), let's try again with NP.Sem having type $\langle \langle e, t \rangle, t \rangle$.

Sam $\lambda P. P(\text{Sam})$ (type raising)
every woman $\lambda P. \forall x. \text{woman}(x) \Rightarrow P(x)$

The appropriate semantic attachment for NP VP is then

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

Semantics of determiners

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

woman	$\lambda x. \text{woman}(x)$	$\langle e, t \rangle$
every	$\lambda Q. \lambda P. \forall x. Q(x) \Rightarrow P(x)$	$\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$
a	$\lambda Q. \lambda P. \exists x. Q(x) \wedge P(x)$	$\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$
NP \rightarrow Det N	$\{ \text{Det.Sem (N.Sem)} \}$	$\langle \langle e, t \rangle, t \rangle$

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

The semantics of “every woman”:

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

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

It had better be $\langle\langle\langle e, t \rangle, t \rangle, \langle e, t \rangle\rangle$.
(A 3rd order function type!)

TV \rightarrow loves $\{\lambda R^{\langle\langle e, t \rangle, t \rangle} . \lambda z^e . R(\lambda w^e . loves(z, w))\}$
VP \rightarrow TV NP $\{TV.Sem(NP.Sem)\}$

To summarize where we've got to:

Grammar III

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

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

Example

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

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

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$
 $\rightarrow_{\beta} \lambda P.\exists x.computer(x) \wedge P(x)$

Example

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

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

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$
 $\rightarrow_{\beta} \lambda P.\exists x.computer(x) \wedge P(x)$

h.a.t. a computer $\dots \rightarrow_{\beta} \dots$
 $\rightarrow_{\beta} \lambda z.\exists x.computer(x) \wedge h_a_t(z, x)$

Example

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

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

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$
 $\rightarrow_{\beta} \lambda P.\exists x.computer(x) \wedge P(x)$

h.a.t. a computer $\dots \rightarrow_{\beta} \dots$
 $\rightarrow_{\beta} \lambda z.\exists x.computer(x) \wedge h_a_t(z, x)$

(whole sentence) $\dots \rightarrow_{\beta} \dots$
 $\rightarrow_{\beta} \forall x.student(x) \Rightarrow \exists y.computer(y) \wedge h_a_t(x, y)$

Example

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

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

a computer $(\lambda Q.\lambda P.\exists x.Q(x) \wedge P(x))(\lambda x.computer(x))$
 $\rightarrow_{\beta} \lambda P.\exists x.computer(x) \wedge P(x)$

h.a.t. a computer $\dots \rightarrow_{\beta} \dots$
 $\rightarrow_{\beta} \lambda z.\exists x.computer(x) \wedge h_a_t(z, x)$

(whole sentence) $\dots \rightarrow_{\beta} \dots$
 $\rightarrow_{\beta} \forall x.student(x) \Rightarrow \exists y.computer(y) \wedge h_a_t(x, y)$

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

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

$\forall x(\text{student}(x) \rightarrow \exists y(\text{computer}(y) \wedge \text{have_access_to}(x, y)))$

Meaning 2

Possibly the same computer for all students

$\exists y(\text{computer}(y) \wedge \forall x(\text{student}(x) \rightarrow \text{have_access_to}(x, y)))$

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

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

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

- 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)

- 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.