Computing Natural Language Semantics

Informatics 2A: Lecture 25

Adam Lopez

15 November 2015

1. Semantic Composition
    - Review: compositionality, lambda expressions, and logical forms
    - Examples
    - Type raising

2. Semantic (Scope) Ambiguity
    - Definition
    - Semantic Scope
    - Approaches to Scope Ambiguity
    - Underspecification: General Idea

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.

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.

$$\begin{array}{ll} \textit{has a bone} & < e, t > \\ \textit{every dog} & << e, t >, t > \end{array}$$

- 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 (λ = 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.

$\lambda x.sleep(x) \;:\; < e, t >$

'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.

$\lambda y.\lambda x.$ love(x,y) $\quad : \quad < e, < e, t >>$
'The function that takes y to
(the function that takes x to the statement love(x,y))'

$\lambda z.\lambda y.\lambda x.$ give(x,y,z) $\quad : \quad < e, < e, < e, t >>>$
'The function that takes z to
(the function that takes y to
(the function that takes x to the statement 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:

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

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

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

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

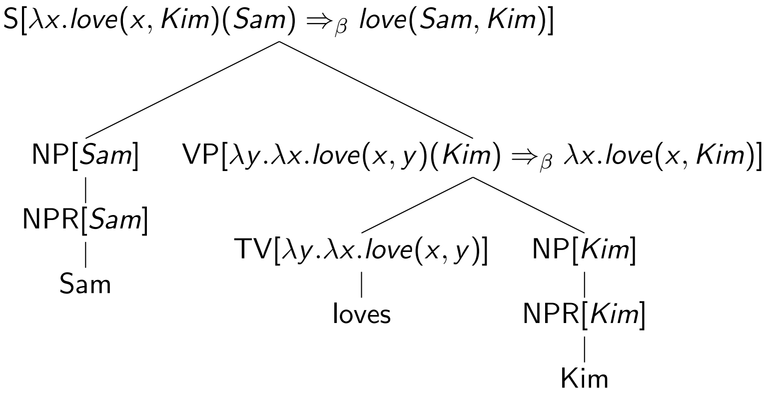
Compositional Semantics: the key idea

Grammar I

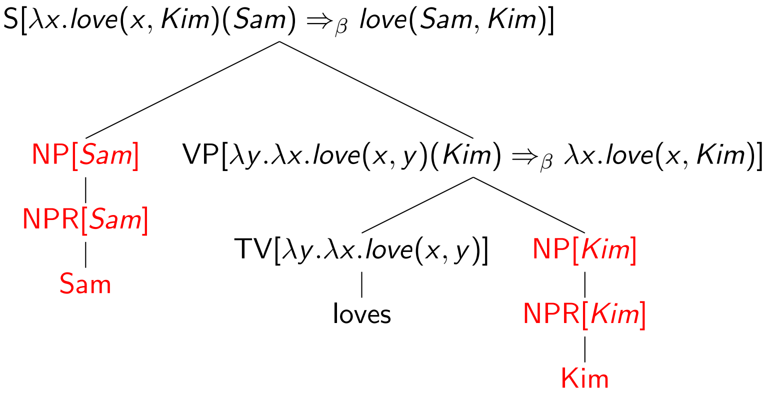
S $\rightarrow$ NP VP	{VP.Sem(NP.Sem)}	t
VP $\rightarrow$ TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
NP $\rightarrow$ NPR	{NPR.Sem}	e
TV $\rightarrow$ loves	{λy.λx.love(x,y)}	$< e, < e, t >>$
NPR $\rightarrow$ Kim	{Kim}	e
NPR $\rightarrow$ Sam	{Sam}	e

- 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.

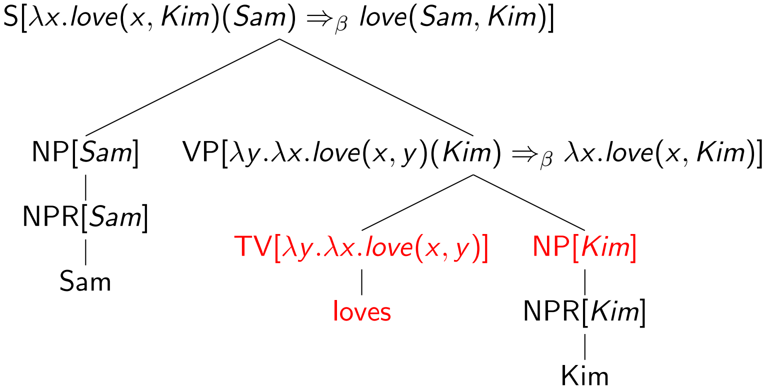
Compositional Semantics: example



Compositional Semantics: example



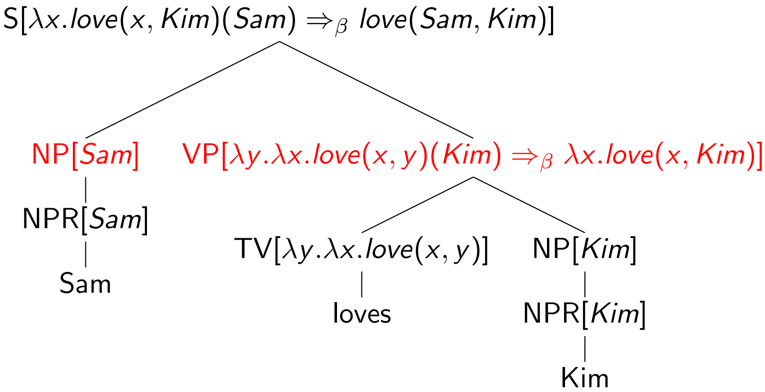
Compositional Semantics: example



Compositional Semantics: example

- $\text{S}[\lambda x.love(x, Kim)(Sam) \Rightarrow_\beta love(Sam, Kim)]$
  - $\text{NP}[Sam]$
    - $\text{NPR}[Sam]$
      - Sam
  - $\text{VP}[\lambda y.\lambda x.love(x, y)(Kim) \Rightarrow_\beta \lambda x.love(x, Kim)]$
    - $\text{TV}[\lambda y.\lambda x.love(x, y)]$
      - loves
    - $\text{NP}[Kim]$
      - $\text{NPR}[Kim]$
        - Kim

Compositional Semantics: example



Compositional Semantics: example

- S[$\lambda x.love(x, Kim)(Sam) \Rightarrow_\beta love(Sam, Kim)$]
  - NP[Sam]
    - NPR[Sam]
      - Sam
  - VP[$\lambda y.\lambda x.love(x, y)(Kim) \Rightarrow_\beta \lambda x.love(x, Kim)$]
    - TV[$\lambda y.\lambda x.love(x, y)$]
      - loves
    - NP[Kim]
      - NPR[Kim]
        - 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	{λx.TV.Sem(x)(NP.Sem)}	$< e, t >$
NP $\rightarrow$ NPR	{NPR.Sem}	e
TV $\rightarrow$ loves	{λx.λy.love(x,y)}	$< e, < e, t >>$
NPR $\rightarrow$ Kim	{Kim}	e
NPR $\rightarrow$ Sam	{Sam}	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:

Sam has access to a computer.
$\exists x(\mathit{computer}(x) \wedge \mathit{have\_access\_to}(\mathit{Sam}, x))$

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

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 $\rightarrow$ Sam	$\{ Sam \}$	e
TV $\rightarrow$ loves	$\{ \lambda y. \lambda x. love(x, y) \}$	$< e, < e, t >>$
N $\rightarrow$ woman	$\{ \lambda z. woman(z) \}$	$< e, t >$
A $\rightarrow$ tall	$\{ \lambda z. tall(z) \}$	$< e, t >$

- 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 $\rightarrow$ 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 $\rightarrow$ Sam	{ Sam }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ woman	{ $\lambda z.woman(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- 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 $\rightarrow$ 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 $\rightarrow$ Sam	{ *Sam* }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ woman	{ $\lambda z.woman(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- 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 $\rightarrow$ 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 $\rightarrow$ Sam	{ *Sam* }	e
TV $\rightarrow$ loves	{ $\lambda y.\lambda x.love(x, y)$ }	$< e, < e, t >>$
N $\rightarrow$ woman	{ $\lambda z.woman(z)$ }	$< e, t >$
A $\rightarrow$ tall	{ $\lambda z.tall(z)$ }	$< e, t >$

- 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?

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

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

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

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

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

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

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

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

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

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

Type raising

- We've given Sam, Kim the semantic type e, and woman the semantic type $< e, t >$.
- 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 $< e, t >$) to get an S.Sem (of type t), let's try again with NP.Sem having type $<< e, t >, t >$.

Sam	$\lambda P.\, P(Sam)$	(type raising)
every woman	$\lambda P.\, \forall x.\, woman(x) \Rightarrow P(x)$	

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 woman’ from that of ’every’ and ’woman’.
- We’ve seen that ‘woman’ has semantic type $< e, t >$, and ’every woman’ 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).

woman	$\lambda x.$ woman(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 woman’ and check that it β-reduces to $\lambda P.\forall x.\, woman(x) \Rightarrow P(x)$.

Example

The semantics of “every woman”:

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!)

TV $\rightarrow$ loves	$\{\lambda R^{<<e,t>,t>}.\lambda z^{e}.\, R(\lambda w^{e}.\, loves(z, w))\}$
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) }	$< e, t >$
NP $\rightarrow$ Sam	{ $\lambda P.P(Sam)$ }	$<< e, t >, t >$
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 $\rightarrow$ woman	{ $\lambda z.woman(z)$ }	$< e, t >$
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’.

$$
\begin{array}{ll}
\text{every student} & (\lambda Q.\lambda P.\forall x.Q(x) \Rightarrow P(x))(\lambda x.student(x)) \\
& \rightarrow_\beta \quad \lambda P.\forall x.\, student(x) \Rightarrow P(x)
\end{array}
$$

Example

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

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

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

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

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

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

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

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

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.