Computing Natural Language Semantics

Informatics 2A: Lecture 26

Mirella Lapata (based on slides by BW, KA, JL)

School of Informatics
University of Edinburgh

17 November 2010

---

1. Semantic Composition for NL
    - Syntax of FOPL
    - Logical Form
2. Semantic (Scope) Ambiguity
    - Definition
    - Semantic Scope
    - Approaches to Scope Ambiguity
3. Underspecification
    - Motivation
    - Underspecification: General Idea

Reading

Required Reading:

- J&M, ch. 18 (Intro → 18.3)
- NLTK book ch. 10 (10.1 → 10.4) `http://nltk.googlecode.com/svn/trunk/doc/book/ch10.html`

Recommended Reading:

- Alexander Koller & Joachim Nieren. *Scope Underspecification and Processing*. ESSLLI 1991 Lecture Notes (pp9–40: general intro to underspecification) `http://www.coli.uni-saarland.de/~koller/papers/esslli99.ps.gz`
- Blackburn & Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. 2005 (ch.1–3)

Syntax of first order predicate logic: summary

This may itself be defined by a CFG (ignore bracketing for now):

$$\begin{array}{lll}
\text{Term} & \rightarrow & \text{Const} \mid \text{Var} \mid \cdots \\
\text{BasicFm} & \rightarrow & \text{UnaryPred (Term)} \\
 & & \mid \text{BinaryPred (Term,Term)} \mid \cdots \\
\text{Fm} & \rightarrow & \text{BasicFm} \mid \neg\text{Fm} \mid \text{Fm}\wedge\text{Fm} \\
 & & \mid \text{Fm}\vee\text{Fm} \mid \text{Fm}\Rightarrow\text{Fm} \\
 & & \mid \forall\ \text{Var . Fm} \mid \exists\ \text{Var . Fm}
\end{array}$$

A formula is called closed if every occurrence of any variable x appears within a quantified formula of the form ∀x.Fm or ∃x.Fm.

Compositional Semantics: the key idea

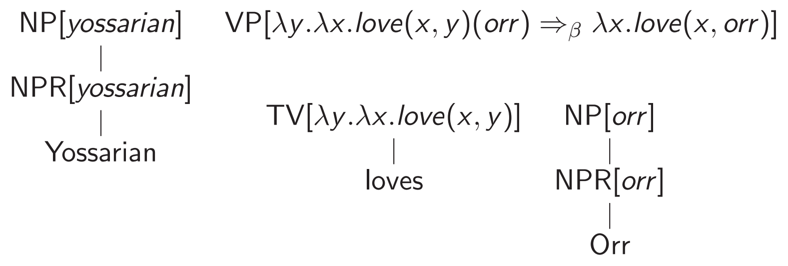
Grammar I

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

- To build a compositional semantics for NL, we attach valuation functions to grammar rules (semantic attachments).
- How to compute the interpretation of the LHS of the rule from the interpretations of its RHS components.
- 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

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



$NP[yossarian]$

$NPR[yossarian]$

Yossarian

$VP[\lambda y.\lambda x.love(x, y)(orr) \Rightarrow_{\beta} \lambda x.love(x, orr)]$

$TV[\lambda y.\lambda x.love(x, y)]$

loves

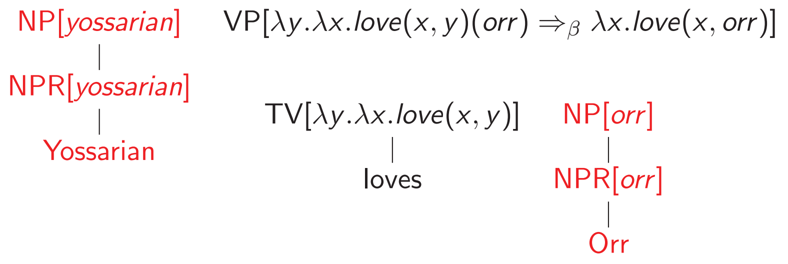
$NP[orr]$

$NPR[orr]$

Orr

Compositional Semantics: example

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



$NP[yossarian]$

$NPR[yossarian]$

Yossarian

$VP[\lambda y.\lambda x.love(x, y)(orr) \Rightarrow_{\beta} \lambda x.love(x, orr)]$

$TV[\lambda y.\lambda x.love(x, y)]$

loves

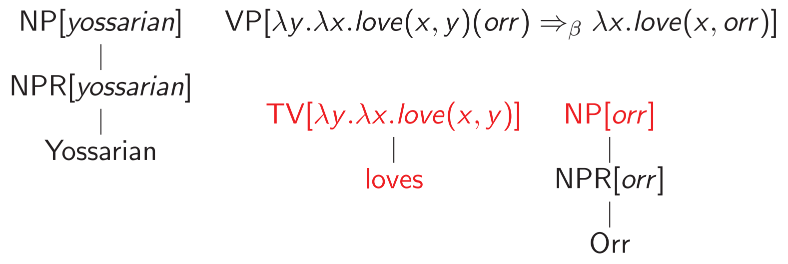
$NP[orr]$

$NPR[orr]$

Orr

Compositional Semantics: example

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



$NP[yossarian]$

$NPR[yossarian]$

Yossarian

$VP[\lambda y.\lambda x.love(x, y)(orr) \Rightarrow_{\beta} \lambda x.love(x, orr)]$

$TV[\lambda y.\lambda x.love(x, y)]$

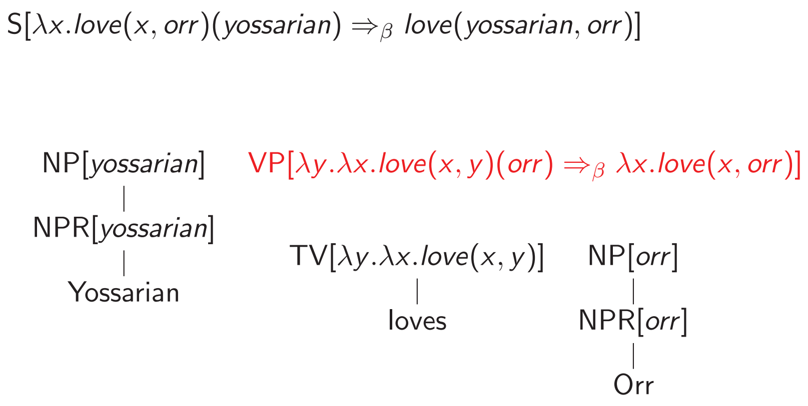
loves

$NP[orr]$

$NPR[orr]$

Orr

Compositional Semantics: example



$S[\lambda x.love(x, orr)(yossarian) \Rightarrow_\beta love(yossarian, orr)]$

$NP[yossarian]$

$NPR[yossarian]$

Yossarian

$VP[\lambda y.\lambda x.love(x, y)(orr) \Rightarrow_\beta \lambda x.love(x, orr)]$

$TV[\lambda y.\lambda x.love(x, y)]$

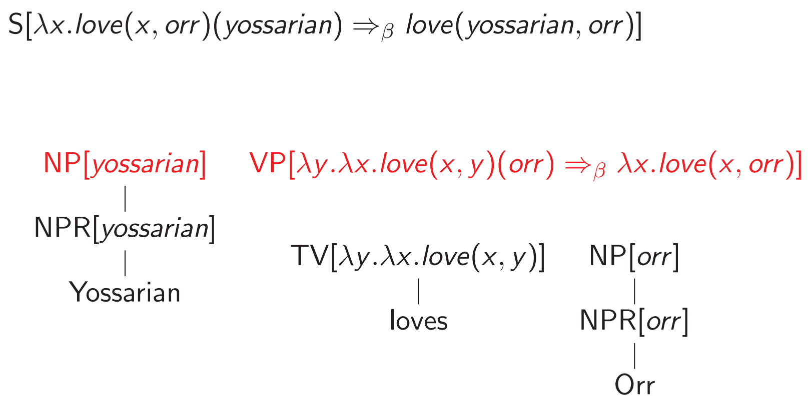
loves

$NP[orr]$

$NPR[orr]$

Orr

Compositional Semantics: example



$S[\lambda x.love(x, orr)(yossarian) \Rightarrow_\beta love(yossarian, orr)]$

$NP[yossarian]$

$NPR[yossarian]$

Yossarian

$VP[\lambda y.\lambda x.love(x, y)(orr) \Rightarrow_\beta \lambda x.love(x, orr)]$

$TV[\lambda y.\lambda x.love(x, y)]$

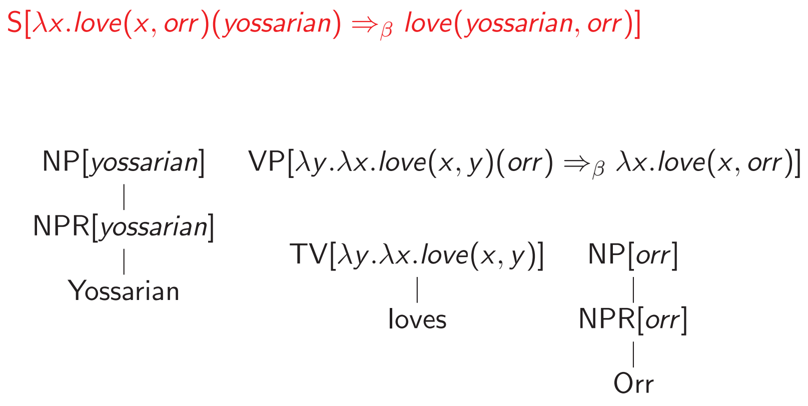
loves

$NP[orr]$

$NPR[orr]$

Orr

Compositional Semantics: example



$S[\lambda x.love(x, orr)(yossarian) \Rightarrow_\beta love(yossarian, orr)]$

$NP[yossarian]$

$NPR[yossarian]$

Yossarian

$VP[\lambda y.\lambda x.love(x, y)(orr) \Rightarrow_\beta \lambda x.love(x, orr)]$

$TV[\lambda y.\lambda x.love(x, y)]$

loves

$NP[orr]$

$NPR[orr]$

Orr

Compositional Semantics, continued

What about the interpretation of an NP other than a proper names whose FOPL interpretation contains 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?

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	$\{\lambda$y.λx.have_access_to(x,y)$\}$	$< e, < e, t >>$
NP → a NOM	$\{\exists x$.NOM.Sem(x)$\}$	$< e, t >$
NP → every NOM	$\{\forall x$.NOM.Sem(x)$\}$	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	$\{\lambda$y.λx.have_access_to(x,y)$\}$	$< e, < e, t >>$
NP → a NOM	$\{\exists x$.NOM.Sem(x)$\}$	$< e, t >$
NP → every NOM	$\{\forall x$.NOM.Sem(x)$\}$	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	$\{\lambda$y.λx.have_access_to(x,y)$\}$	$< e, < e, t >>$
NP → a NOM	$\{\exists x$.NOM.Sem(x)$\}$	$< e, t >$
NP → every NOM	$\{\forall x$.NOM.Sem(x)$\}$	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	$\{\lambda$y.λx.have_access_to(x,y)$\}$	$< e, < e, t >>$
NP → a NOM	$\{\exists x$.NOM.Sem(x)$\}$	$< e, t >$
NP → every NOM	$\{\forall x$.NOM.Sem(x)$\}$	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	{λy.λx.have_access_to(x,y)}	$< e, < e, t >>$
NP → a NOM	{∃x.NOM.Sem(x)}	$< e, t >$
NP → every NOM	{∀x.NOM.Sem(x)}	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	{λy.λx.have_access_to(x,y)}	$< e, < e, t >>$
NP → a NOM	{∃x.NOM.Sem(x)}	$< e, t >$
NP → every NOM	{∀x.NOM.Sem(x)}	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

- We want to get interpretations for a computer and every student from the above syntactic rules and semantic attachments.

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	{λy.λx.have_access_to(x,y)}	$< e, < e, t >>$
NP → a NOM	{∃x.NOM.Sem(x)}	$< e, t >$
NP → every NOM	{∀x.NOM.Sem(x)}	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

- We want to get interpretations for a computer and every student from the above syntactic rules and semantic attachments.
- This is nonsensical as it stands: NOM.Sem has type e, but the expression $\exists x$.NOM.Sem(x) requires it to have type $< e, t >$.

Grammar II

S → NP VP	{VP.Sem(NP.Sem)}	t
VP → TV NP	{TV.Sem(NP.Sem)}	$< e, t >$
TV → has access to	{λy.λx.have_access_to(x,y)}	$< e, < e, t >>$
NP → a NOM	{∃x.NOM.Sem(x)}	$< e, t >$
NP → every NOM	{∀x.NOM.Sem(x)}	$< e, t >$
NPR → John	{john}	e
NOM → N	{NOM.Sem}	e
N → student	{student}	e
N → computer	{computer}	e

- We want to get interpretations for a computer and every student from the above syntactic rules and semantic attachments.
- This is nonsensical as it stands: NOM.Sem has type e, but the expression $\exists x$.NOM.Sem(x) requires it to have type $< e, t >$.
- In addition, the sentence ‘Every student has access to a computer’ is somewhat ambiguous (scoping ambiguity).

Type raising (Cf. Tutorial Sheet 8, part 2)

- The first problem seems to arise from our decision that NP.Sem should have type e.
- 'john' is an entity — but which entity is 'every student'?
- 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	$\lambda P.\, P(john)$
every student	$\lambda P.\, \forall x.\, student(x) \Rightarrow P(x)$

The appropriate semantic attachment for NP VP is then

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

Semantics of determiners

- Using this approach, we can also derive the semantics of 'every student' from that of 'every' and 'student'.
- Whereas proper nouns (*e.g.* John) denote entities (e), common nouns (*e.g.* student) should denote *properties* of entities ($< e, t >$).
- *Determiners* (*e.g.* every, a, no, not every) should therefore have interpretations of type $<< e, t >, << e, t >, t >>$.

student	$\lambda x.$ student(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 student' and check that it β-reduces to what we had before.

More on type raising

- Recall the grammar rule: 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

- Recall the grammar rule: 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 $\rightarrow$ has access to	$\{\lambda R^{<<e,t>,t>}.\lambda z^{e}.\, R(\lambda w^{e}.\, h\_a\_t(z, w))\}$
VP $\rightarrow$ TV NP	{TV.Sem(NP.Sem)}

Example

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

Example

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

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

Example

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

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

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.

Clicker Questions

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. $(\lambda P.\forall y.\exists x.P(y,x))(\lambda x\lambda y.L(x,y))$

What does the sentence *Every student has access to a laptop* mean?

1. Every student has a different laptop
2. Every student has the same laptop
3. Both (1) and (2)

Semantic Ambiguity

While the sentence 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 laptop per student
$\forall x(student(x) \rightarrow \exists y(laptop(y) \wedge have\_access\_to(x,y)))$

Meaning 2

Possibly the same laptop for all students
$\exists y(laptop(y) \wedge \forall x(student(y) \rightarrow have\_access\_to(x,y)))$

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.

Coping with Scope: options

1. **Enumerate all interpretations**. Computationally unattractive!
2. **Store the interpretation of sub-units** (as in chart parsing). Empty the stores after the whole sentence is parsed. The order of emptying the stores determines what has scope over what. (See `nltk.sem.cooper_storage`.)
3. Use an **underspecified representation** that can be further specified to each of the multiple interpretations.

Why use underspecification? (1)

Constraints from the discourse or the outside world may get us directly to the intended interpretation, rather than needing to select from among enumerated alternatives:

Every student has access to a laptop. The European Research Foundation just donated 200 new laptops for use in Inf2a. (⇒ Meaning 1)

Every student has access to a laptop. It can be borrowed from the ITO. (⇒ Meaning 2)

Motivating Underspecification (2)

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

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.