Intensional independence without world variables in LFG+Glue

Matthew Gotham
University of Oxford

25th International LFG Conference
23–26 June 2020

Outline

Introduction: intensionality with an ‘s’

Outline

Outline

Outline

Introduction: intensionality with an ‘s’

Extensional and intensional contexts

(1) Elizabeth is Queen of the UK and Harald is King of Norway $\leftrightarrow$ Elizabeth is Queen of Australia and Harald is King of Norway .

Extensional context: substitution of coreferential expressions preserves truth value.

Extensional and intensional contexts

(1) Elizabeth is Queen of the UK and Harald is King of Norway $\leftrightarrow$ Elizabeth is Queen of Australia and Harald is King of Norway .

Extensional context: substitution of coreferential expressions preserves truth value.

(2) Malcolm knows/believes/recognizes/cares that Elizabeth is Queen of the UK. $\not\leftrightarrow$ Malcolm knows/believes/recognizes/cares that Elizabeth is Queen of Australia.

Intensional context: substitution of coreferential expressions does not necessarily preserve truth value.

Examples of intensional contexts

- Propositional attitudes

  Malcolm thinks/hopes/fears/wishes that ____.

Examples of intensional contexts

- Propositional attitudes

  Malcolm thinks/hopes/fears/wishes that ____.
- Modals

  ____ is possible/necessary/obligatory/permitted/obvious.

Examples of intensional contexts

- Propositional attitudes

  Malcolm thinks/hopes/fears/wishes that ____.
- Modals

  ____ is possible/necessary/obligatory/permitted/obvious.
- Counterfactual conditionals

  If ____, then Sarah would be better known.

Possible worlds semantics

The mainstream approach:

	Extension	Intension
Sentence	Truth value	Function from possible worlds to truth values (*proposition*)
Name	Entity	Function from possible worlds to entities (*individual concept*)

(Non-specific) *de re* / *de dicto*

(3) Anna wants a left-handed player to win.

(Non-specific) *de re* / *de dicto*

(3) Anna wants a left-handed player to win.

a. Specific *de re*:

$\Rightarrow$ $\exists x.\mathrm{LHP}' x w_@ \wedge \mathrm{want}'\mathrm{anna}'(\mathrm{win}' x) w_@$

(Non-specific) *de re* / *de dicto*

(3) Anna wants a left-handed player to win.

a. Specific *de re*:

$\Rightarrow$ $\exists x.\mathrm{LHP}' x w_{@} \wedge \mathrm{want}' \mathrm{anna}' (\mathrm{win}' x) w_{@}$

b. Non-specific *de re*:

$\Rightarrow$ $\mathrm{want}' \mathrm{anna}' (\lambda w.\exists x.\mathrm{LHP}' x w_{@} \wedge \mathrm{win}' x w) w_{@}$

(Non-specific) *de re* / *de dicto*

(3) Anna wants a left-handed player to win.

a. Specific *de re*:

$\Rightarrow$ $\exists x.\mathrm{LHP}'x w_{@} \wedge \mathrm{want}'\mathrm{anna}'(\mathrm{win}'x)w_{@}$

b. Non-specific *de re*:

$\Rightarrow$ $\mathrm{want}'\mathrm{anna}'(\lambda w.\exists x.\mathrm{LHP}'x w_{@} \wedge \mathrm{win}'x w)w_{@}$

c. *de dicto*:

$\Rightarrow$ $\mathrm{want}'\mathrm{anna}'(\lambda w.\exists x.\mathrm{LHP}'x w \wedge \mathrm{win}'x w)w_{@}$

(Where $w_{@}$ denotes the actual world)

Background

Background

Two theories of intensionality

The Binding Theory of Intensionality (BTI)

The Binding Theory of Intensionality (BTI)

- Certain lexical items combine with world (or situation) 'pronouns' in the syntax.

The Binding Theory of Intensionality (BTI)

- Certain lexical items combine with world (or situation) ‘pronouns’ in the syntax.
- Like personal pronouns, they are interpreted as variables (but over worlds not individuals).

The Binding Theory of Intensionality (BTI)

- Certain lexical items combine with world (or situation) ‘pronouns’ in the syntax.
- Like personal pronouns, they are interpreted as variables (but over worlds not individuals).
- Intensional status is determined by the level at which these pronouns are ‘bound’.

(3) Anna wants a left-handed player to win.

(3) Anna wants a left-handed player to win.

de dicto

[Anna [wants [Σ_1 [[[a *pro*$_1$] left-handed player] to win]]]]

non-specific *de re*

[Σ_1 [Anna [wants [[[a *pro*$_1$] left-handed player] to win]]]]

specific *de re*

[Σ_1 [[[a *pro*$_1$] left-handed player]$_2$ [Anna [wants [t$_2$ to win]]]]]

(after Schwarz 2012)

The Scope Theory of Intensionality (STI)

The intensional status of an expression is determined by its scope relative to expressions that create intensional contexts, e.g.

- propositional attitude verbs,
- modal predicates,
- conditionals.

(3) Anna wants a left-handed player to win.

de dicto

[Anna [wants [[a left-handed player] to win]]]

specific *de re*

[[a left-handed player]$_1$ [Anna [wants [t$_1$ to win]]]]

non-specific *de re*?

(3) Anna wants a left-handed player to win.

de dicto

[Anna [wants [[a left-handed player] to win]]]

specific *de re*

[[a left-handed player]$_1$ [Anna [wants [t$_{1:e}$ to win]]]]

non-specific *de re*

[Anna [wants [[a left-handed player]$_1$ [$^{\triangle}$ [t$_{1:e}$ to win]]]]]

(Keshet 2011)

[[a left-handed player]$_1$ [Anna [wants [t$_{1:(e\to t)\to t}$ to win]]]]

(von Fintel & Heim 2011)

Background

Desiderata for a theory of intensionality

Unavailable *de re*

(Romoli & Sudo 2009)

(4) John wants to meet the wife of the President.

$\Rightarrow$ $\text{want}'\text{john}'(\lambda w.\text{meet}'\text{john}'\iota x.\text{wife-of}'(\iota y.\text{president}' y w) x w) w_@$

$\Rightarrow$ $\text{want}'\text{john}'(\lambda w.\text{meet}'\text{john}'\iota x.\text{wife-of}'(\iota y.\text{president}' y w_@) x w_@) w_@$

$\Rightarrow$ $\text{want}'\text{john}'(\lambda w.\text{meet}'\text{john}'\iota x.\text{wife-of}'(\iota y.\text{president}' y w_@) x w) w_@$

$\not\Rightarrow$ $\text{want}'\text{john}'(\lambda w.\text{meet}'\text{john}'\iota x.\text{wife-of}'(\iota y.\text{president}' y w) x w_@) w_@$

Another unavailable *de re*

(5) Mr. Smith thinks that Lucy wrote every essay that Tim wrote.

$\Rightarrow$ $\text{think}'\text{smith}'(\lambda w.\forall z.\text{ETM}' z w_{@} \rightarrow \text{write}'\text{lucy}' w) w_{@}$

Another unavailable *de re*

(5) Mr. Smith thinks that Lucy wrote every essay that Tim wrote.

$\Rightarrow$ $\text{think}'\text{smith}'(\lambda w.\forall z.\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w)w_{@}$

(6) Mr. Smith thinks that Tim should get detention because Lucy wrote every essay that he/Tim wrote.

$$\not\Rightarrow \text{think}'\text{smith}'\big(\text{because}'(\lambda w.\forall z.(\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w)) (\text{should}'\text{tim-detention}')\big)w_{@}$$

Depth of embedding is not the issue

(7) The Principal knows that Mr. Smith thinks that Lucy wrote every essay that Tim wrote.

Depth of embedding is not the issue

(7) The Principal knows that Mr. Smith thinks that Lucy wrote every essay that Tim wrote.

$\Rightarrow$ $\text{know}'\text{principal}'(\text{think}'\text{smith}'(\lambda w.\forall z.\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w))w_{@}$

Independence from quantificational scope

(Keshet 2010, after Bäuerle 1983)

(8) George thinks every Red Sox player is staying in some five-star hotel downtown.

$\Rightarrow \text{think}'\text{george}'(\lambda w.\exists x.\text{hotel}' x w \wedge \forall y.\text{RSP}' y w_{@} \rightarrow \text{stay-in}' x y w) w_{@}$

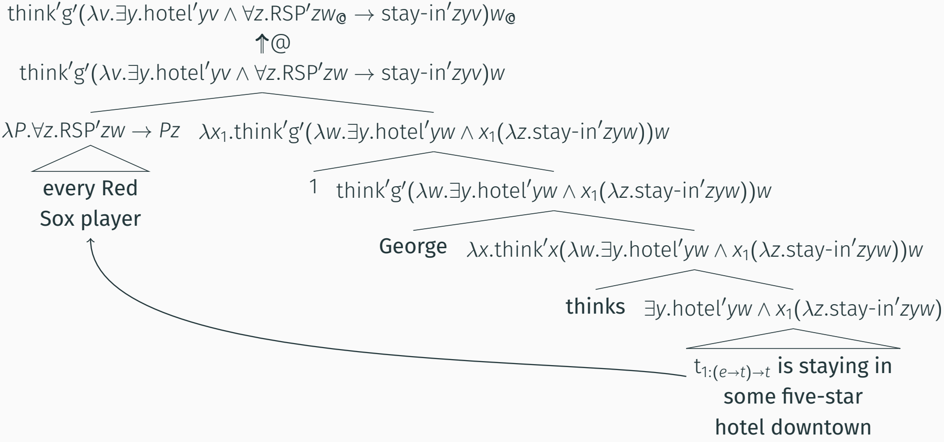
In this reading, *every Red Sox player* is in the scope of *some five-star hotel downtown*, which is intensionally dependent on *thinks*, but *every Red Sox player* is intensionally independent of *thinks*.

$\exists > \forall$
$\text{think}' > \exists$
$\forall > \text{think}'$

Background

The von Fintel & Heim (2011) suggestion

Higher-order traces



Reflections on the suggestion

- Use of a higher-type trace means that the QNP can move above the point of intensionalization so as to be interpreted *de re*, while its quantificational force ‘reconstructs’ back to its base position. So we get intensional independence.

Reflections on the suggestion

- Use of a higher-type trace means that the QNP can move above the point of intensionalization so as to be interpreted *de re*, while its quantificational force ‘reconstructs’ back to its base position. So we get intensional independence.
- This general idea will form the basis of my own proposal.

Reflections on the suggestion

- Use of a higher-type trace means that the QNP can move above the point of intensionalization so as to be interpreted *de re*, while its quantificational force ‘reconstructs’ back to its base position. So we get intensional independence.
- This general idea will form the basis of my own proposal.
- However, within the framework of transformational syntax this supposes covert movement from out of a finite clause, which is supposed to be disallowed.

- What von Fintel & Heim would have to say is that some types of movement are allowed or not depending on the type of trace left behind.

- What von Fintel & Heim would have to say is that some types of movement are allowed or not depending on the type of trace left behind.
- They would also have to propose a novel constraint on covert movement to account for unavailable *de re* in the *essay*-type cases.

- What von Fintel & Heim would have to say is that some types of movement are allowed or not depending on the type of trace left behind.
- They would also have to propose a novel constraint on covert movement to account for unavailable *de re* in the *essay*-type cases.
- In contrast, as we'll see, the LFG framework gives us the resources to state the necessary distinctions neatly.

An LFG+Glue approach

What ‘without world variables’ means

What ‘without world variables’ means

- For legibility’s sake I will actually use world variables in the semantic representation—unlike in the abstract, which made use of $^{\wedge}$ and $^{\vee}$ operators (Montague 1973).

What ‘without world variables’ means

- For legibility’s sake I will actually use world variables in the semantic representation—unlike in the abstract, which made use of $^\wedge$ and $^\vee$ operators (Montague 1973).
- Use of $^\wedge$ and $^\vee$ instead constrains the expressibility of the meaning representation language in certain ways.

What ‘without world variables’ means

- For legibility’s sake I will actually use world variables in the semantic representation—unlike in the abstract, which made use of $^\wedge$ and $^\vee$ operators (Montague 1973).
- Use of $^\wedge$ and $^\vee$ instead constrains the expressibility of the meaning representation language in certain ways.
- I will instead show that these constraints are met by using only a single world variable name: *w* (Zimmermann 1989).

What ‘without world variables’ means

- For legibility’s sake I will actually use world variables in the semantic representation—unlike in the abstract, which made use of $^\wedge$ and $^\vee$ operators (Montague 1973).
- Use of $^\wedge$ and $^\vee$ instead constrains the expressibility of the meaning representation language in certain ways.
- I will instead show that these constraints are met by using only a single world variable name: *w* (Zimmermann 1989).
- For legibility’s sake I will show the binding patterns of occurrences of *w* with colours. These colours have no theoretical status.

Glue semantics crash course

(9) Jim smiles.

$$f : \begin{bmatrix} \textsc{pred} & \text{`smile'} \\ \textsc{subj} & g : [\ \text{``Jim''}\ ] \end{bmatrix}$$

$$\textit{Jim} \rightsquigarrow \text{jim}' : \uparrow$$

$$\textit{smiles} \rightsquigarrow \text{smile}' : (\uparrow \textsc{subj}) \multimap \uparrow$$

(9) Jim smiles.

$$f : \begin{bmatrix} \text{PRED} & \text{`smile'} \\ \text{SUBJ} & g : [\text{``Jim''}] \end{bmatrix}$$

$$\textit{Jim} \rightsquigarrow \text{jim}' : g$$

$$\textit{smiles} \rightsquigarrow \text{smile}' : g \multimap f$$

(9) Jim smiles.

$$f : \begin{bmatrix} \text{PRED} & \text{`smile'} \\ \text{SUBJ} & g : [\ \text{``Jim''}\ ] \end{bmatrix}$$

$$\textit{Jim} \rightsquigarrow \text{jim}' : g$$

$$\textit{smiles} \rightsquigarrow \text{smile}' : g \multimap f$$

$$\frac{\text{smile}' : g \multimap f \qquad \text{jim}' : g}{\text{smile}'\text{jim}' : f} \multimap_E$$

(10) A police officer guards every exit.

$\Rightarrow\ \exists x.\mathrm{officer}'x \wedge \forall y.\mathrm{exit}'y \rightarrow \mathrm{guard}'xy$ (surface scope)

$\Rightarrow\ \forall y.\mathrm{exit}'y \rightarrow \exists x.\mathrm{officer}'x \wedge \mathrm{guard}'xy$ (inverse scope)

Scope ambiguity

(10) A police officer guards every exit.

$\Rightarrow\ \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy$ (surface scope)

$\Rightarrow\ \forall y.\text{exit}'y \rightarrow \exists x.\text{officer}'x \wedge \text{guard}'xy$ (inverse scope)

$$f:\begin{bmatrix} \text{PRED} & \text{‘guard’} \\ \text{SUBJ} & g:\begin{bmatrix} \text{PRED} & \text{‘police officer’} \\ \text{SPEC} & i:\begin{bmatrix} \text{PRED} & \text{‘a’} \end{bmatrix} \end{bmatrix} \\ \text{OBJ} & h:\begin{bmatrix} \text{PRED} & \text{‘exit’} \\ \text{SPEC} & j:\begin{bmatrix} \text{PRED} & \text{‘every’} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Multiple proofs

$$
\begin{aligned}
a &\rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx \\
&\quad : ((\textsc{spec}\uparrow) \multimap \uparrow) \multimap (((\textsc{spec}\uparrow) \multimap \%A) \multimap \%A) \\
&\quad \%A = (\textsc{path}\uparrow) \\
\textit{police officer} &\rightsquigarrow \text{officer}' : (\textsc{spec}\uparrow) \multimap \uparrow \\
\textit{guards} &\rightsquigarrow \text{guard}' : (\uparrow \textsc{subj}) \multimap ((\uparrow \textsc{obj}) \multimap \uparrow) \\
\textit{every} &\rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy \\
&\quad : ((\textsc{spec}\uparrow) \multimap \uparrow) \multimap (((\textsc{spec}\uparrow) \multimap \%B) \multimap \%B) \\
&\quad \%B = (\textsc{path}\uparrow) \\
\textit{exit} &\rightsquigarrow \text{exit}' : (\textsc{spec}\uparrow) \multimap \uparrow
\end{aligned}
$$

Multiple proofs

$$
\begin{aligned}
a &\rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx \\
&\quad : (g \multimap i) \multimap ((g \multimap f) \multimap f) \\
&\quad \%A := f \\
\textit{police officer} &\rightsquigarrow \text{officer}' : g \multimap i \\
\textit{guards} &\rightsquigarrow \text{guard}' : g \multimap (h \multimap f) \\
\textit{every} &\rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy \\
&\quad : (h \multimap j) \multimap ((h \multimap f) \multimap f) \\
&\quad \%B := f \\
\textit{exit} &\rightsquigarrow \text{exit}' : h \multimap j
\end{aligned}
$$

Surface scope interpretation

$$
\dfrac{
\dfrac{\text{a}' : (g \multimap i) \multimap ((g \multimap f) \multimap f) \qquad \text{officer}' : g \multimap i}{(g \multimap f) \multimap f}
\qquad
\dfrac{\dfrac{\dfrac{\text{every}' : (h \multimap j) \multimap ((h \multimap f) \multimap f) \qquad \text{exit}' : h \multimap j}{(h \multimap f) \multimap f} \qquad \dfrac{\text{guard}' : g \multimap (h \multimap f) \qquad [g]^1}{h \multimap f}}{f}}{g \multimap f}\ {\multimap_{I,1}}
}{
\begin{array}{c}
\text{a}'\text{officer}'(\lambda x.\text{every}'\text{exit}'(\text{guard}'x)) : f \\
\equiv \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy : f
\end{array}
}
$$

Inverse scope interpretation

$$
\dfrac{
\dfrac{\begin{array}{c}\text{every}' : \\ (h \multimap j) \multimap \\ ((h \multimap f) \multimap f)\end{array} \quad \begin{array}{c}\text{exit}' : \\ h \multimap j\end{array}}{(h \multimap f) \multimap f}
\quad
\dfrac{
\dfrac{
\dfrac{\begin{array}{c}\text{a}' : \\ (g \multimap i) \multimap \\ ((g \multimap f) \multimap f)\end{array} \quad \begin{array}{c}\text{officer}' : \\ g \multimap i\end{array}}{(g \multimap f) \multimap f}
\quad
\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{guard}' : \\ g \multimap (h \multimap f)\end{array} \quad [g]^1}{h \multimap f} \quad [h]^2}{f}}{g \multimap f} \multimap I, 1
}{f}
}{h \multimap f} \multimap I, 2
}{
\text{every}'\text{exit}'(\lambda y.\text{a}'\text{officer}'(\lambda x.\text{guard}' xy)) : f
}
$$

$$
\equiv \forall y.\text{exit}' y \rightarrow \exists x.\text{officer}' x \wedge \text{guard}' xy : f
$$

An LFG+Glue approach

Upgrading for intensional independence

The type of determiners

Standard extensional type

$$(e \to t) \to (e \to t) \to t$$

The type of determiners

Standard extensional type

$$(e \to t) \to (e \to t) \to t$$

Add argument positions for worlds (intensionalize)

$$(e \to s \to t) \to (e \to s \to t) \to s \to t$$

The type of determiners

Standard extensional type

$$(e \to t) \to (e \to t) \to t$$

Add argument positions for worlds (intensionalize)

$$(e \to s \to t) \to (e \to s \to t) \to s \to t$$

Abbreviate: let $p := s \to t$

$$(e \to p) \to (e \to p) \to p$$

The type of determiners

Standard extensional type

$$(e \to t) \to (e \to t) \to t$$

Add argument positions for worlds (intensionalize)

$$(e \to s \to t) \to (e \to s \to t) \to s \to t$$

Abbreviate: let $p := s \to t$

$$(e \to p) \to (e \to p) \to p$$

Type-raise the second argument position

$$(e \to p) \to (((e \to p) \to p) \to p) \to p$$

Two layers of scope-taking

$$(e \to p) \to (((e \to p) \to p) \to p) \to p, \qquad \text{where } p := s \to t$$

determiner D

$(\uparrow \text{PRED}) = \text{`det'}$

$\%A = (\text{QUANT\_SCOPE\_PATH} \uparrow)$

$\%B = (\text{INT\_SCOPE\_PATH} \uparrow)$

$\lambda F.\lambda V.\lambda w^s.\exists P^{e \to t}.P = (\lambda x.Fxw) \wedge V(\lambda G.\lambda w.\text{det}' P(\lambda x.Gyw))w$

$: [(\text{SPEC} \uparrow) \multimap \uparrow] \multimap [((((\text{SPEC} \uparrow) \multimap \%A) \multimap \%A) \multimap \%B) \multimap \%B]$

Where $F, G :: e \to p$ and $V :: ((e \to p) \to p) \to p$.

When quantificational and intensional scope are the same

Let $\uparrow := q, (\textsc{spec}\uparrow) := e$ and $\%A := \%B := p$.

When quantificational and intensional scope are the same

Let $\uparrow := q, (\textsc{spec}\uparrow) := e$ and $\%A := \%B := p$. Then

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \begin{array}{c}[\mathit{determiner}] : \\ (e \multimap q) \multimap ((((e \multimap p) \multimap p) \multimap p) \multimap p)\end{array}
        \quad [F : e \multimap q]^1
      }{
        [\mathit{determiner}]F : (((e \multimap p) \multimap p) \multimap p) \multimap p
      }
      \quad
      \dfrac{
        \dfrac{[V : (e \multimap p) \multimap p]^2 \quad [G : e \multimap p]^3}{VG : p}
      }{
        \lambda V.VG : ((e \multimap p) \multimap p) \multimap p
      }\; 2
    }{
      [\mathit{determiner}]F(\lambda V.VG) : p
    }
  }{
    \lambda G.[\mathit{determiner}]F(\lambda V.VG) : (e \multimap p) \multimap p
  }\; 3
}{
  \lambda F.\lambda G.[\mathit{determiner}]F(\lambda V.VG) : (e \multimap q) \multimap ((e \multimap p) \multimap p)
}\; 1
$$

When quantificational and intensional scope are the same

Let $\uparrow := q, (\textsc{spec}\uparrow) := e$ and $\%A := \%B := p$. Then

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\begin{matrix}[\mathit{determiner}] : \\ (e \multimap q) \multimap ((((e \multimap p) \multimap p) \multimap p) \multimap p)\end{matrix} \quad [F : e \multimap q]^1}{[\mathit{determiner}]F : (((e \multimap p) \multimap p) \multimap p) \multimap p}
      \quad
      \dfrac{\dfrac{[V : (e \multimap p) \multimap p]^2 \quad [G : e \multimap p]^3}{VG : p}}{\lambda V.VG : ((e \multimap p) \multimap p) \multimap p}\,2
    }{[\mathit{determiner}]F(\lambda V.VG) : p}
  }{\lambda G.[\mathit{determiner}]F(\lambda V.VG) : (e \multimap p) \multimap p}\,3
}{\lambda F.\lambda G.[\mathit{determiner}]F(\lambda V.VG) : (e \multimap q) \multimap ((e \multimap p) \multimap p)}\,1
$$

$$
\equiv \lambda F.\lambda G.\lambda w.\exists P.P = (\lambda x.Fxw) \wedge \mathrm{det}' P(\lambda x.Gxw) : (e \multimap q) \multimap ((e \multimap p) \multimap p)
$$

When quantificational and intensional scope are the same

Let $\uparrow := q$, $(\textsc{spec}\uparrow) := e$ and $\%A := \%B := p$. Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}[\mathit{determiner}]:\\ (e \multimap q) \multimap ((((e \multimap p) \multimap p) \multimap p) \multimap p)\end{array} \quad [F : e \multimap q]^1}{[\mathit{determiner}]F : (((e \multimap p) \multimap p) \multimap p) \multimap p} \quad \dfrac{\dfrac{[V : (e \multimap p) \multimap p]^2 \quad [G : e \multimap p]^3}{VG : p}}{\lambda V.VG : ((e \multimap p) \multimap p) \multimap p}\,2}{[\mathit{determiner}]F(\lambda V.VG) : p}}{\lambda G.[\mathit{determiner}]F(\lambda V.VG) : (e \multimap p) \multimap p}\,3}{\lambda F.\lambda G.[\mathit{determiner}]F(\lambda V.VG) : (e \multimap q) \multimap ((e \multimap p) \multimap p)}\,1$$

$$\equiv \lambda F.\lambda G.\lambda w.\exists P.P = (\lambda x.Fxw) \wedge \mathrm{det}'P(\lambda x.Gxw) : (e \multimap q) \multimap ((e \multimap p) \multimap p)$$

$$\equiv \lambda F.\lambda G.\lambda w.\mathrm{det}'(\lambda x.Fxw)(\lambda x.Gxw) : (e \multimap q) \multimap ((e \multimap p) \multimap p)$$

When they're not

(8) George thinks every Red Sox player is staying in some five-star hotel downtown.

$$\Rightarrow \text{think}'\text{george}'(\lambda w.\exists x.\text{hotel}'xw \wedge \forall y.\text{RSP}'yw_{@} \rightarrow \text{stay-in}'xyw)w_{@}$$

$$f:\left[\begin{array}{ll}
\text{PRED} & \text{`think'} \\
\text{TENSE} & \text{PRES} \\
\text{SUBJ} & g:[\ \text{“George”}\ ] \\
\text{COMP} & h:\left[\begin{array}{ll}
\text{PRED} & \text{`stay'} \\
\text{TENSE} & \text{PRES} \\
\text{SUBJ} & i:\left[\begin{array}{ll}
\text{PRED} & \text{`player'} \\
\text{SPEC} & j:\left[\begin{array}{ll}\text{PRED} & \text{`every'}\end{array}\right]
\end{array}\right] \\
\text{OBL}_{\text{LOC}} & k:[\ \text{“in some five-star hotel downtown”}\ ]
\end{array}\right]
\end{array}\right]$$

Meaning constructors

think $\rightsquigarrow$ $\text{think}' : (\uparrow \textsc{subj}) \multimap (\uparrow \textsc{comp}) \multimap \uparrow$

George $\rightsquigarrow$ $\text{george}' : \uparrow$

Red Sox player $\rightsquigarrow$ $\text{RSP}' : \uparrow \multimap (\uparrow \textsc{spec})$

stay in $\rightsquigarrow$ $\text{stay-in}' : (\uparrow \textsc{subj}) \multimap (\uparrow \textsc{obl}_{\textsc{loc}}) \multimap \uparrow$

a hotel $\rightsquigarrow$

$[\textit{a hotel}] := \lambda U.\lambda w.\exists z.\text{hotel}' zw \wedge Uzw : (\uparrow \multimap \%A) \multimap \%A$

every $\rightsquigarrow$

$[\textit{every}] := \lambda F.\lambda V.\lambda w.\exists P.P = (\lambda x.Fxw) \wedge V(\lambda G.\lambda w.\forall z.Pz \rightarrow Gzw)w$

$: ((\textsc{spec} \uparrow) \multimap \uparrow) \multimap ((((\textsc{spec} \uparrow) \multimap \%A) \multimap \%A) \multimap \%B) \multimap \%B$

$think \rightsquigarrow \mathrm{think}' : g \multimap h \multimap f$

$George \rightsquigarrow \mathrm{george}' : g$

$Red\ Sox\ player \rightsquigarrow \mathrm{RSP}' : i \multimap j$

$stay\ in \rightsquigarrow \text{stay-in}' : i \multimap k \multimap h$

$in\ a\ hotel \rightsquigarrow$

$[in\ a\ hotel] := \lambda U.\lambda w.\exists z.\mathrm{hotel}' zw \wedge Uzw : (k \multimap h) \multimap h$

$every \rightsquigarrow$

$$[every] := \lambda F.\lambda V.\lambda w.\exists P.P = (\lambda x.Fxw) \wedge V(\lambda G.\lambda w.\forall z.Pz \rightarrow Gzw)w$$
$$: (i \multimap j) \multimap (((i \multimap h) \multimap h) \multimap f) \multimap f$$

The intended interpretation

$$
\dfrac{
\begin{array}{c} \vdots \\ [\mathit{every}]\text{RSP}' : \\ (((i \multimap h) \multimap h) \multimap f) \multimap f \end{array}
\qquad
\dfrac{
\dfrac{
\begin{array}{c} \vdots \\ \text{think}'\text{george}' : \\ h \multimap f \end{array}
\quad
\dfrac{
\begin{array}{c} [\mathit{a\ hotel}] : \\ (k \multimap h) \multimap h \end{array}
\quad
\begin{array}{c} \left[\begin{array}{c} G : \\ (i \multimap h) \multimap h \end{array}\right]^1 \quad \begin{array}{c} \text{stay-in}' : \\ i \multimap k \multimap h \end{array} \\ \vdots \\ \lambda v.G(\lambda u.\text{stay-in}'uv) : k \multimap h \end{array}
}{
[\mathit{a\ hotel}](\lambda v.G(\lambda u.\text{stay-in}'uv)) : h
}
}{
\text{think}'\text{george}'([\mathit{a\ hotel}](\lambda v.G(\lambda u.\text{stay-in}'uv))) : f
}
}{
\begin{array}{c} \lambda G.\text{think}'\text{george}'([\mathit{a\ hotel}](\lambda v.G(\lambda u.\text{stay-in}'uv))) : \\ ((i \multimap h) \multimap h) \multimap f \end{array}
}\;1
}{
[\mathit{every}]\text{RSP}'(\lambda G.\text{think}'\text{george}'([\mathit{a\ hotel}](\lambda v.G(\lambda u.\text{stay-in}'uv)))) : f
}
$$

The intended interpretation

$$\left[\begin{array}{c} G : \\ (i \multimap h) \multimap h \end{array}\right]^1 \qquad \begin{array}{c} \text{stay-in}' : \\ i \multimap k \multimap h \end{array}$$

$$\dfrac{\dfrac{\begin{array}{c} \vdots \\ \text{think}'\text{george}' : \\ h \multimap f \end{array} \qquad \dfrac{\begin{array}{c} [a\ hotel] : \\ (k \multimap h) \multimap h \end{array} \qquad \begin{array}{c} \vdots \\ \lambda v.G(\lambda u.\text{stay-in}'uv) : k \multimap h \end{array}}{[a\ hotel](\lambda v.G(\lambda u.\text{stay-in}'uv)) : h}}{\dfrac{\text{think}'\text{george}'([a\ hotel](\lambda v.G(\lambda u.\text{stay-in}'uv))) : f}{\begin{array}{c}\lambda G.\text{think}'\text{george}'([a\ hotel](\lambda v.G(\lambda u.\text{stay-in}'uv))) : \\ ((i \multimap h) \multimap h) \multimap f\end{array}}\, 1}}{}$$

$$\dfrac{\begin{array}{c} \vdots \\ [every]\text{RSP}' : \\ (((i \multimap h) \multimap h) \multimap f) \multimap f \end{array} \qquad \begin{array}{c}\lambda G.\text{think}'\text{george}'([a\ hotel](\lambda v.G(\lambda u.\text{stay-in}'uv))) : \\ ((i \multimap h) \multimap h) \multimap f\end{array}}{[every]\text{RSP}'(\lambda G.\text{think}'\text{george}'([a\ hotel](\lambda v.G(\lambda u.\text{stay-in}'uv)))) : f}$$

$$\equiv \lambda w.\exists P.P = (\lambda x.\text{RSP}'xw)$$

$$\wedge\ \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.Py \rightarrow \text{stay-in}'yzw)w : f$$

$$\lambda w.\exists P.P = (\lambda x.\text{RSP}'xw)$$
$$\wedge\ \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.Py \rightarrow \text{stay-in}'yzw)w$$

$$\lambda w.\exists P.P = (\lambda x.\text{RSP}'xw) \\ \wedge \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.Py \to \text{stay-in}'yzw)w$$

$$\Downarrow @$$

$$\exists P.P = (\lambda x.\text{RSP}'xw_@) \\ \wedge \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.Py \to \text{stay-in}'yzw)w_@$$

$$\lambda w.\exists P.P = (\lambda x.\text{RSP}'xw)$$

$$\wedge\ \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.Py \rightarrow \text{stay-in}'yzw)w$$

$$\Downarrow @$$

$$\exists P.P = (\lambda x.\text{RSP}'xw_{@})$$

$$\wedge\ \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.Py \rightarrow \text{stay-in}'yzw)w_{@}$$

$$\equiv \text{think}'\text{george}'(\lambda w.\exists z.\text{hotel}'zw \wedge \forall y.\text{RSP}'yw_{@} \rightarrow \text{stay-in}'yzw)w_{@}$$

An LFG+Glue approach

Formalising the constraints

The nature of QUANT_SCOPE_PATH

- Generalization: an expression cannot take quantificational scope outside its minimal finite clause.

The nature of QUANT_SCOPE_PATH

- Generalization: an expression cannot take quantificational scope outside its minimal finite clause.
- Solution:

$$\text{QUANT\_SCOPE\_PATH} := \left(\begin{array}{ccc} \text{GGF}^{*} & \text{GF} & \text{SPEC} \\ \neg(\rightarrow \text{TENSE}) & & \end{array} \right)$$

John wants to meet the wife of the President

$$
f : \left[\begin{array}{ll}
\text{PRED} & \text{`want'} \\
\text{TENSE} & \text{PRES} \\
\text{SUBJ} & g : [\ \text{``John''}\ ] \\
\text{XCOMP} & h : \left[\begin{array}{ll}
\text{PRED} & \text{`meet'} \\
\text{SUBJ} & \\
\text{OBJ} & i : \left[\begin{array}{ll}
\text{PRED} & \text{`wife'} \\
\text{SPEC} & j : \left[\begin{array}{ll}\text{PRED} & \text{`the'}\end{array}\right] \\
\text{OBL}_{\text{GEN}} & k : \left[\begin{array}{ll}
\text{PRED} & \text{`president'} \\
\text{SPEC} & l : \left[\begin{array}{ll}\text{PRED} & \text{`the'}\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

The STI and embedding

- All scope theories, including this one, predict the unavailability of the ‘wife’ *de re*, ‘president’ *de dicto* reading.

The STI and embedding

- All scope theories, including this one, predict the unavailability of the ‘wife’ *de re*, ‘president’ *de dicto* reading.
- The reason is that *the president* is embedded within *the wife of the president*, so attempting to get the latter to scope strictly wider than the former inevitably leaves unbound variables, hence an improper derivation.

*John wants to meet [the wife of [the President]]

The nature of INT_SCOPE_PATH

(6) Mr. Smith thinks that Tim should get detention because Lucy wrote every essay that he/Tim wrote.

$$\not\Rightarrow \text{think}'\text{smith}'\big(\text{because}'(\lambda w.\forall z.(\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w))$$
$$(\text{should}'\text{tim-detention}')\big)w_{@}$$

(7) The Principal knows that Mr. Smith thinks that Lucy wrote every essay that Tim wrote.

$$\Rightarrow \text{know}'\text{principal}'\big(\text{think}'\text{smith}'\big(\lambda w.\forall z.\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w\big)\big)w_{@}$$

(6) Mr. Smith thinks that Tim should get detention because Lucy wrote every essay that he/Tim wrote.

$$\not\Rightarrow \text{think}'\text{smith}'(\text{because}'(\lambda w.\forall z.(\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w))(\text{should}'\text{tim-detention}'))w_{@}$$

(7) The Principal knows that Mr. Smith thinks that Lucy wrote every essay that Tim wrote.

$$\Rightarrow \text{know}'\text{principal}'(\text{think}'\text{smith}'(\lambda w.\forall z.\text{ETM}'zw_{@} \rightarrow \text{write}'\text{lucy}'w))w_{@}$$

Generalization: a nominal predicate can be interpreted *de re* from within two finite clauses, but not from within an ADJUNCT island within a finite clause.

$$
f:\begin{bmatrix}
\text{PRED} & \text{`think'} \\
\text{TENSE} & \text{PRES} \\
\text{SUBJ} & [\ \text{``Mr. Smith''}\ ] \\
\text{COMP} & g:\begin{bmatrix}
\text{PRED} & \text{`should'} \\
\text{TENSE} & \text{PRES} \\
\text{SUBJ} & [\ \text{``Tim''}\ ] \\
\text{XCOMP} & [\ \text{``get detention''}\ ] \\
\text{ADJ} & \left\{ h:\begin{bmatrix}
\text{PRED} & \text{`because'} \\
\text{OBJ} & i:\begin{bmatrix}
\text{PRED} & \text{`write'} \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & [\ \text{``Lucy''}\ ] \\
\text{OBJ} & j:\begin{bmatrix}
\text{PRED} & \text{`essay'} \\
\text{SPEC} & k:\begin{bmatrix}\text{PRED} & \text{`every'}\end{bmatrix} \\
\text{ADJ} & \left\{[\ \text{``Tim wrote''}\ ]\right\}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \right\}
\end{bmatrix}
\end{bmatrix}
$$

$$f:\begin{bmatrix} \text{PRED} & \text{‘think’} \\ \text{COMP} & g:\begin{bmatrix} \text{PRED} & \text{‘should’} \\ \text{TENSE} & \text{PRES} \\ \text{ADJ} & \left\{ h:\begin{bmatrix} \text{PRED} & \text{‘because’} \\ \text{OBJ} & i:\begin{bmatrix} \text{PRED} & \text{‘write’} \\ \text{OBJ} & j:\begin{bmatrix} \text{PRED} & \text{‘essay’} \\ \text{SPEC} & k:\begin{bmatrix} \text{PRED} & \text{‘every’} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix}$$

- Task: permit (INT_SCOPE_PATH k) $= g, h$ or i but not f.

$$f:\left[\begin{array}{ll}\textsc{pred} & \text{`think'}\\ \textsc{comp} & g:\left[\begin{array}{ll}\textsc{pred} & \text{`should'}\\ \textsc{tense} & \textsc{pres}\\ \textsc{adj} & \left\{h:\left[\begin{array}{ll}\textsc{pred} & \text{`because'}\\ \textsc{obj} & i:\left[\begin{array}{ll}\textsc{pred} & \text{`write'}\\ \textsc{obj} & j:\left[\begin{array}{ll}\textsc{pred} & \text{`essay'}\\ \textsc{spec} & k:\left[\textsc{pred}\ \ \text{`every'}\right]\end{array}\right]\end{array}\right]\end{array}\right]\right\}\end{array}\right]\end{array}\right]$$

- Task: permit $(\textsc{int\_scope\_path}\ k) = g, h$ or i but not f.
- Solution:

$$\textsc{int\_scope\_path} := \left(\begin{array}{ccccc}\textsc{ggf}^* & (\textsc{adj} \in) & \textsc{ggf}^* & \textsc{gf} & \textsc{spec}\\ \neg(\rightarrow \textsc{tense}) & & & & \end{array}\right)$$

$$f:\left[\begin{array}{ll} \text{PRED} & \text{‘think’} \\ \text{COMP} & g:\left[\begin{array}{ll} \text{PRED} & \text{‘should’} \\ \text{TENSE} & \text{PRES} \\ \text{ADJ} & \left\{ h:\left[\begin{array}{ll} \text{PRED} & \text{‘because’} \\ \text{OBJ} & i:\left[\begin{array}{ll} \text{PRED} & \text{‘write’} \\ \text{OBJ} & j:\left[\begin{array}{ll} \text{PRED} & \text{‘essay’} \\ \text{SPEC} & k:\left[\begin{array}{ll}\text{PRED} & \text{‘every’}\end{array}\right]\end{array}\right]\end{array}\right]\end{array}\right]\right\}\end{array}\right]\end{array}\right]$$

- Task: permit $(\text{INT\_SCOPE\_PATH}\ k) = g, h$ or i but not f.
- Solution:

$$\text{INT\_SCOPE\_PATH} := \left(\begin{array}{ccccc} \text{GGF}^* & (\text{ADJ} \in) & \text{GGF}^* & \text{GF} & \text{SPEC} \\ \neg(\rightarrow \text{TENSE}) & & & & \end{array} \right)$$

- Result: $(f\ \text{COMP})$ (i.e. g) has a tense value and $\text{ADJ} \in$ is ungovernable, hence there's no instance of INT_SCOPE_PATH such that $(\text{INT\_SCOPE\_PATH}\ k) = f$.

De re from within two finite clauses

$$f : \begin{bmatrix} \text{PRED} & \text{`know'} \\ \text{SUBJ} & [\ \text{“the Principal”}\ ] \\ \text{COMP} & \begin{bmatrix} \text{PRED} & \text{`think'} \\ \text{SUBJ} & [\ \text{“Mr. Smith”}\ ] \\ \text{COMP} & \begin{bmatrix} \text{PRED} & \text{`write'} \\ \ldots & \\ \text{SUBJ} \mid \text{SPEC} & g : \begin{bmatrix} \text{PRED} & \text{`every'} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- INT_SCOPE_PATH :=

$$\left(\begin{array}{cccccc} \text{GGF}^* & (\text{ADJ} \in) & \text{GGF}^* & \text{GF} & \text{SPEC} \\ \neg(\rightarrow \text{TENSE}) & & & & \end{array} \right)$$

$$f : \begin{bmatrix} \textsc{pred} & \text{`know'} \\ \textsc{subj} & [\ \text{``the Principal''}\ ] \\ \textsc{comp} & \begin{bmatrix} \textsc{pred} & \text{`think'} \\ \textsc{subj} & [\ \text{``Mr. Smith''}\ ] \\ \textsc{comp} & \begin{bmatrix} \textsc{pred} & \text{`write'} \\ \ldots & \\ \textsc{subj}\,|\,\textsc{spec} & g : \begin{bmatrix} \textsc{pred} & \text{`every'} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- INT_SCOPE_PATH :=
$$\left(\begin{array}{ccccc} \textsc{ggf}^{*} & (\textsc{adj} \in) & \textsc{ggf}^{*} & \textsc{gf} & \textsc{spec} \\ \neg(\rightarrow \textsc{tense}) & & & & \end{array} \right)$$
- Result: $(f\ \textsc{comp}\ \textsc{comp}\ \textsc{subj}\ \textsc{spec}) = g.$ ✓

Conclusion

Discussion

- I have proposed a method to rectify the undergeneration inherent in existing versions of the STI.

Discussion

- I have proposed a method to rectify the undergeneration inherent in existing versions of the STI.
- The method involves raising the type of determiner lexical entries to give them two, potentially distinct, scope positions: quantificational and intensional.

Discussion

- I have proposed a method to rectify the undergeneration inherent in existing versions of the STI.
- The method involves raising the type of determiner lexical entries to give them two, potentially distinct, scope positions: quantificational and intensional.
- This method is reminiscent of a suggestion from von Fintel & Heim (2011), but is not hamstrung by a conflict with a pre-existing syntactic theory of locality.

Discussion

- I have proposed a method to rectify the undergeneration inherent in existing versions of the STI.
- The method involves raising the type of determiner lexical entries to give them two, potentially distinct, scope positions: quantificational and intensional.
- This method is reminiscent of a suggestion from von Fintel & Heim (2011), but is not hamstrung by a conflict with a pre-existing syntactic theory of locality.
- In fact, the LFG+Glue architecture gives us just the tools we need to state the right, independent, constraints on quantificational and intensional scope.

Thanks!

This research is funded by the

LEVERHULME TRUST

References

See the accompanying handout.