Overview

Modularity

Semantic Parts

This Talk
What it is

understand a complex phenomenon by

• factoring it into simple(r) parts
• analysing these parts

entire phenomenon = combination of parts
Linguistics

Complex Phenomenon
ability to use language

Simple(r) Parts

- Phonetics
- Phonology
- Morphology
- Syntax
- Semantics
- Pragmatics
Semantics

Complex Phenomenon
Had a politician pushed the issue, he would have been arrested

Simple(r) Parts?

- argument structure
- scope
- intentionality
- dynamics
- context sensitivity
- tense
A basic idea

- semantic objects are λ-terms
- a part specifies how to ’update’ a λ-term
  - so that it reflects the structure of the part
An intentionality module
a function $\text{INT}$ which turns

- a non-intentional meaning
- into an intentional one

$$\text{INT}(\text{no(\text{student})}(\lambda x.\text{can(\text{laugh}(x))})) =$$
$$\lambda w.\text{no(\text{student}(w))}(\lambda x.\exists w'. w \ R \ w' \ \land \ (\text{laugh}(w')(x \ w'))))$$
The Ideal

1. identify a domain
2. specify non-predictable meanings
   • most things are predictable!
3. specify how to generate predictable meanings
   only works if compositional:
   • meanings of whole
   • determined by meanings of parts
Homomorphisms

**Strings**
replace a symbol

• with a string

**Trees**
replace a symbol with \( k \) daughters

• with a tree with \( k \) empty leaves

**\( \lambda \)-Terms**
replace a symbol of one type

• with a term of similar type
we want to view simpler expressions

\[
\text{every(kitten)} : (e \to t) \to t
\]

as ’abbreviating’ more complex ones

\[
\text{every(kitten)} : (e \to t) \to t
\]

in a compositional way:

\[
\text{every(kitten)} = \text{every(kitten)}
\]

\[
(e \to t) \to t = (\bar{e} \to \bar{t}) \to \bar{t}
\]
Type homomorphism

specify what each atomic type 'abbreviates':

\[ h(c) = \alpha \]

extend this homomorphically to complex types:

\[ \overline{c} = h(c) \]

\[ \alpha \rightarrow \beta = \overline{\alpha} \rightarrow \overline{\beta} \]

**Single type semantics**
both \( e \) and \( t \) abbreviate \( q \) (with \( D_q = D_{(et)t} \))

\[ e \rightarrow e \rightarrow t = \overline{e} \rightarrow \overline{e} \rightarrow \overline{t} = q \rightarrow q \rightarrow q \]
Lambda homomorphisms

specify what each constant ’abbreviates’:

\[ h(k) = M \]

extend this homomorphically to complex terms:

\[ \overline{k} = h(k) \]
\[ \overline{x} = x \]
\[ \overline{\lambda x. M} = \lambda x. \overline{M} \]
\[ \overline{M \ N} = \overline{M} \ \overline{N} \]

You keep the structure of the term and just replace constants with their definitions
To make sense, we require that lambda homomorphisms and type homomorphisms come in pairs:

**This means that**

- you replace a symbol $c : \alpha$  
- with a term $\overline{c} : \overline{\alpha}$  
  - of ’similar’ type
I will show how this works

• analyse context-sensitivity including dynamic binding
• decompose this into two independent modules
  • CON implementing context-sensitivity
  • DYN implementing (static) dynamicity
• discuss options
  • implementing instead dynamic dynamicity (à la DMG)
  • extensions to RST/SDRT
Dynamics

Discourse

Modularizing Dynamics

Example

Discourse Structure
The dynamics of pronoun reference

Sentences can set up discourse referents for other sentences

model this as scope:

1. \( S(\ldots T(\ldots) \ldots) \)
   - \( T \) can access discourse refs introduced by \( S \)
2. \( S(\ldots) \land T(\ldots) \)
   - \( T \) cannot access discourse refs introduced by \( S \)

the primitive notion here is sentence scope

  - motivated by context sensitivity
  - formally independent thereof
Interpreting sentences in discourse

Basic observation
context grows throughout the discourse

Formal implementation
sentences in a discourse scope over each other

\[[S. T.] = [S] \circ [T] = \lambda x. [S]( [T](x)) \]
Dynamicization

On types

\[ \bar{e} = e \]
\[ \bar{t} = t \rightarrow t \]

The intuition sentences scope over the rest of the discourse
Inherent Dynamicity

*and*

is internally and externally dynamic

*some*

is internally and externally dynamic

*if... then*

is internally dynamic

*DETs*

are internally dynamic
Break dynamicization into two steps

1. internal dynamicity
2. external dynamicity
Inherent Internal dynamicity

Determiners and Conservativity

strong

\[
\text{int}(\text{det}_S) := \lambda P, Q. \text{det} P (P \rightarrow Q)
\]

weak

\[
\text{int}(\text{det}_W) := \lambda P, Q. \text{det} P (P \land Q)
\]

Implication and Classical Equivalence

\[
\text{impl}(\text{if...then}) := \lambda \phi, \psi. \neg (\phi \land \neg \psi)
\]

Accounts for Internal Dynamicity!
Inherent external dynamicity

*and* is externally dynamic
\[
\text{ext}(\text{and}) := \lambda \Phi, \Psi, \psi. \Phi(\Psi \psi) = \mathbb{B}
\]

*some* is externally dynamic
\[
\text{ext}(\text{some}) := \lambda P, Q, \psi. \text{some}(\lambda x. P x \top)(\lambda x. Q x \psi)
\]
Lambda homomorphisms

dyn
ext \circ \text{impl} \circ \text{int}
Lambda homomorphisms

\[
\text{dyn} \circ \text{ext} \circ \text{impl} \circ \text{int}
\]

\[
\begin{align*}
\text{int}(\det_w) &= \lambda P, Q.\det P (P \land Q) \\
\text{int}(\det_s) &= \lambda P, Q.\det P (P \rightarrow Q) \\
\text{int}(k) &= k
\end{align*}
\]
**Lambda homomorphisms**

\[ \text{dyn} \circ \text{ext} \circ \text{impl} \circ \text{int} \]

**int**

\[
\text{int}(\det_w) = \lambda P, Q. \det P (P \land Q)
\]

\[
\text{int}(\det_s) = \lambda P, Q. \det P (P \rightarrow Q)
\]

\[
\text{int}(k) = k
\]

**impl**

\[
\text{impl}(\text{if...then}) = \lambda \phi, \psi. \neg (\phi \land \neg \psi)
\]

\[
\text{impl}(k) = k
\]
Lambda homomorphisms

dyn
ext \circ \text{impl} \circ \text{int}

int
\text{int}(\text{det}_W) = \lambda P, Q. \text{det} P (P \land Q)
\text{int}(\text{det}_S) = \lambda P, Q. \text{det} P (P \rightarrow Q)
\text{int}(k) = k

impl
\text{impl}(\text{if...then}) = \lambda \phi, \psi. \neg (\phi \land \neg \psi)
\text{impl}(k) = k

ext
\text{ext}(\text{and}) = \mathbb{B}
\text{ext}(\text{some}) = \lambda P, Q, \phi. \text{some}(\lambda x. P(x)(\top))(\lambda x. Q(x)(\phi))
\text{ext}(k) = \text{DYN}(k)
Dynamic Lifting

Intrinsically static expressions are predictable

\[ \text{DYN}_\alpha : \alpha \rightarrow \overline{\alpha} \]

\[ \text{DYN}_e(a) := a \]

\[ \text{DYN}_t(\phi) := \lambda \psi. \phi \land \psi \]

\[ \text{DYN}_{\alpha\beta}(f) := \text{DYN}_\beta \circ f \circ \text{STA}_\alpha \]

\[ = \lambda A. \text{DYN}_\beta(f (\text{STA}_\alpha A)) \]
Dynamic Lifting

Intrinsically static expressions are predictable

\[ \text{STA}_\alpha : \overline{\alpha} \rightarrow \alpha \]

\[ \text{STA}_e(A) := A \]

\[ \text{STA}_t(\Phi) := \Phi \top \]

\[ \text{STA}_{\alpha\beta}(F) := \text{STA}_\beta \circ F \circ \text{DYN}_\alpha \]

\[ = \lambda a.\text{STA}_\beta(F (\text{DYN}_\alpha a)) \]
Examples

- $\text{Dyn}_{tt}(\text{not}) := \lambda \Phi, \psi. \neg (\Phi \top) \land \psi$
- $\text{Dyn}_{eet}(\text{praise}) := \lambda x, y, \psi. \text{praise} x y \land \psi$
- $\text{Dyn}_{(et)(et)t}(\text{every}) := \lambda P, Q, \psi. \text{every}(\lambda x. P x \top)(\lambda x. Q x \top) \land \psi$
An example

Start with:

\[
\begin{align*}
\text{a} : (et)(et)t & \quad \text{boy} : et \\
\text{jump} : et & \quad \text{laugh} : et \\
\text{and} : ttt & \quad \text{he} : e
\end{align*}
\]

Let

\[
\phi = \text{and}(\text{a(boy)(jump)})(\text{laugh(he)}): t
\]

Then

\[
\text{dyn}(\phi) : t \rightarrow t
\]

Where

\[
\text{dyn}(\phi) \equiv \lambda \psi. \text{a(boy)}(\lambda x. \text{boy}(x) \land \text{laugh(he)} \land \psi)
\]
Discourse Relations

1. John had a lovely evening
2. He had a great meal
3. He ate salmon
4. He devoured cheese
5. He won a dancing competition

Elaborating vs Narrating

• 2 and 5 elaborate on 1
• 2 and 5 are a narrative
• 3 and 4 elaborate on 2
• 3 and 4 are a narrative
Discourse Structure

ELAB

had lovely

evening

NARR

won dance-
ing comp

ELAB

had great

meal

NARR

ate  devoured

salmon  cheese
Pronouns can only refer to certain referents introduced in the previous discourse tree

1. start at rightmost leaf
2. walk anywhere except down a left branch of NARR relation
Example

1. John had a lovely evening
2. He had a great meal
3. He ate salmon
4. He devoured cheese
5. He won a dancing competition
6. # It was nice and pink
Interpreting Relations

Subordinating
\[ [\text{ELAB}] = \lambda \Phi, \Psi, \phi. \Phi(\Psi \phi) \]

Coordinating
\[ [\text{NARR}] = \lambda \Phi, \Psi, \phi. (\Phi \top) \land (\Psi \phi) \]
Interpreting Relations

Subordinating
\[\llbracket \text{ELAB} \rrbracket = \lambda \Phi, \Psi, \phi. \Phi(\Psi \phi)\]

Coordinating
\[\llbracket \text{NARR} \rrbracket = \lambda \Phi, \Psi, \phi. (\Phi \top) \land (\Psi \phi)\]

Right Frontier Constraint
Pronouns can only refer to certain referents introduced in the previous discourse tree

1. start at rightmost leaf
2. walk anywhere except down a left branch of NARR relation
Summary

- can study logic of discourse
  - independently of context-sensitivity
Context Sensitivity

Overview

Pronouns as variables

Pronouns as identity functions

Identity functions vs variables

Pronouns as definites

Synthesis

Modularizing Context Sensitivity

Example

Rethinking contexts
we now want to study context-sensitivity

• independently of discourse dynamics
What is $[[\text{he}]]$?

- variable $x_7$
- id func $\lambda x.x$
- definite $\text{the}(\lambda x.\text{boy}(x) \land \text{near}(x)(\text{kim}))$
All theories agree in main respects:

*The first thought that a completely semantically naive person might have is that a pronoun just picks out in a discourse context some contextually salient individual.*  

(Jacobson 2015)

just disagree in

1. what contexts are
2. how you pick out individuals
• why this is the case:
  • pronouns as variables
  • pronouns as identity functions
  • pronouns as definites

• a synthesis
• a pronoun, he, denotes a variable, $x_7$
PRONOUNS denote variables

- a pronoun, he, denotes a variable, $x_7$
- infinitely many variables ($x_1, x_2, x_3, \ldots$) and so pronouns are infinitely ambiguous
PRONOUNS denote variables

- A pronoun, he, denotes a variable, $x_7$
- Infinitely many variables ($x_1, x_2, x_3, \ldots$) and so pronouns are infinitely ambiguous

Alternatively: push ambiguity from interface into lexicon
PRONOUNS denote variables

• a pronoun, he, denotes a variable, \( x_7 \)
• infinitely many variables \((x_1, x_2, x_3, \ldots)\) and so pronouns are infinitely ambiguous

**alternatively:** push ambiguity from interface into lexicon

• \( h_{e1}, h_{e2}, h_{e3}, \ldots \)
PRONOUNS denote variables

- a pronoun, *he*, denotes a variable, \(x_7\)
- infinitely many variables \((x_1, x_2, x_3, \ldots)\) and so pronouns are infinitely ambiguous

**alternatively:** push ambiguity from interface into lexicon

- \(\text{he}_1, \text{he}_2, \text{he}_3, \ldots\)
- pronoun resolution is *choosing the right disambiguation*
PRONOUNS denote variables

• a pronoun, he, denotes a variable, \( x_7 \)
• infinitely many variables \((x_1, x_2, x_3, \ldots)\) and so pronouns are infinitely ambiguous

alternatively: push ambiguity from interface into lexicon

• \( he_1, he_2, he_3, \ldots \)

• pronoun resolution is choosing the right disambiguation

• syntax says next to nothing about how to disambiguate pronouns
Pronouns denote VARIABLES

• what does it mean to ’denote’ a variable???
• look at the semantics of the \( \lambda \) calculus

\[
\llbracket \Gamma \vdash x^\alpha \rrbracket = \lambda g.g(x)
\]

\[
\llbracket \Gamma \vdash (M^\alpha \rightarrow^\beta N^\alpha)^\beta \rrbracket = \lambda g.\llbracket \Gamma \vdash M^\alpha \rightarrow^\beta \rrbracket g (\llbracket \Gamma \vdash N^\alpha \rrbracket g)
\]

\[
\llbracket \Gamma \vdash (\lambda x^\alpha.M^\beta)^\alpha \rightarrow^\beta \rrbracket = \lambda g, y.\llbracket \Gamma \vdash M^\beta \rrbracket g[x := y]
\]
Pronouns denote VARIABLES

• what does it mean to ’denote’ a variable???
• look at the semantics of the \( \lambda \) calculus

\[
\llbracket \Gamma \vdash x^\alpha \rrbracket = \lambda g. g(x)
\]

\[
\llbracket \Gamma \vdash (M^{\alpha \rightarrow \beta} N^\alpha)^\beta \rrbracket = \lambda g. \llbracket \Gamma \vdash M^{\alpha \rightarrow \beta} \rrbracket g (\llbracket \Gamma \vdash N^\alpha \rrbracket g)
\]

\[
\llbracket \Gamma \vdash (\lambda x^\alpha. M^\beta)^{\alpha \rightarrow \beta} \rrbracket = \lambda g, y. \llbracket \Gamma \vdash M^\beta \rrbracket g[x := y]
\]
Pronouns as variables

We write:

\[ [x_i]^g = g_i \]

This really means:

\[ [x_i] = \lambda g.g_i \]

And so:

\[ x_i : \gamma \rightarrow e \]

Compositionality?

Having pronouns denote variables is as directly compositional as anything else
PRONOUNS denote identity functions

- a pronoun, he, denotes the identity function $\lambda x.x$
- *just one* identity function, so no ambiguity
- **pronoun resolution** is *resolving syntactic ambiguity*
  - expressions keep track of contained pronouns
  - contexts are broken up and passed to the correct daughter
- syntax **completely determines** reference
  - John said that Bill likes his mother is 3 ways syntactically ambiguous
Pronouns denote IDENTITY FUNCTIONS

- we look at a resource sensitive $\lambda$ calculus

\[
\begin{align*}
\llbracket x^\alpha \vdash x^\alpha \rrbracket &= \lambda p.p \\
\llbracket \Gamma, \Delta \vdash (M^\alpha \rightarrow^\beta N^\alpha)^\beta \rrbracket &= \lambda p. \text{let} (p_1, p_2) := \text{splitAt } |\Gamma| \ p \ \text{in} \\
&\quad \llbracket \Gamma \vdash M^\alpha \rightarrow^\beta \rrbracket \ p_1 (\llbracket \Delta \vdash N^\alpha \rrbracket \ p_2) \\
\llbracket \Gamma \vdash (\lambda x^\alpha . M^\beta)^\alpha \rightarrow^\beta \rrbracket &= \lambda p, y. \llbracket \Gamma, x^\alpha \vdash M^\beta \rrbracket (p ++ y)
\end{align*}
\]
Pronouns denote IDENTITY FUNCTIONS

- we look at a resource sensitive \( \lambda \) calculus

\[
\begin{align*}
\llbracket x^\alpha \vdash x^\alpha \rrbracket &= \lambda p. p \\
\llbracket \Gamma, \Delta \vdash (M^\alpha \rightarrow^\beta N^\alpha)^\beta \rrbracket &= \lambda p. \text{let} (p_1, p_2) := \text{splitAt } |\Gamma| \ p \ \text{in} \\
& \quad \llbracket \Gamma \vdash M^\alpha \rightarrow^\beta \rrbracket \ p_1 (\llbracket \Delta \vdash N^\alpha \rrbracket \ p_2) \\
\llbracket \Gamma \vdash (\lambda x^\alpha.M^\beta)^\alpha \rightarrow^\beta \rrbracket &= \lambda p, y. \llbracket \Gamma, x^\alpha \vdash M^\beta \rrbracket (p ++ y)
\end{align*}
\]
Pronouns denote IDENTITY FUNCTIONS

• we look at a resource sensitive \( \lambda \) calculus

\[
\begin{align*}
\llbracket x^\alpha \vdash x^\alpha \rrbracket &= \lambda p.p \\
\llbracket \Gamma, \Delta \vdash (M^\alpha \rightarrow^\beta N^\alpha)^\beta \rrbracket &= \lambda p.\text{let } (p_1, p_2) := \text{splitAt } |\Gamma| \ p \ in \\
&\quad \llbracket \Gamma \vdash M^\alpha \rightarrow^\beta \rrbracket \ p_1 \ (\llbracket \Delta \vdash N^\alpha \rrbracket \ p_2) \\
\llbracket \Gamma \vdash (\lambda x^\alpha . M^\beta)^\alpha \rightarrow^\beta \rrbracket &= \lambda p, y.\llbracket \Gamma, x^\alpha \vdash M^\beta \rrbracket (p ++ y)
\end{align*}
\]
Contexts

- Both require sentences to be interpreted with respect to a sequence of individuals
  - represents the context in which the sentence was uttered
- Assignment functions are very poor representations of context!
  - which assignment represents this context?

\[
\gamma := \bigcup_{n=0}^{\infty} E^n
\]

variables
\[
\gamma = E^N
\]

- Pronouns are defined only on contexts of size 1
Resolving reference in context

- Pronouns are resolved to an individual in a context
- Very poor strategies for resolving reference

\[
\text{variables } \llbracket \text{he}_7 \rrbracket (\langle c_1, c_2, \ldots \rangle) = c_7
\]

'pick the seventh thing in the context’

\[
\text{id } \llbracket \text{he} \rrbracket (\langle c \rangle) = c
\]

'pick the only thing in the context’
Pronouns as paraphrases

- pronouns mean definite descriptions
  - either because they syntactically are dds
  - or because they mean the same thing as a dd
- dds pick out an individual in a ‘minimal situation’
- situations are everywhere!!!
A Synthesis

• a pronoun denotes a function of type $\gamma \rightarrow e$
  • i.e. pronoun resolution algorithms
• $\gamma$ is the type of a context

This is:

• just what we’ve been doing all along,
• except without *a priori* commitments to the metaphysics of contexts
\( \gamma \) is the type of a context

A context can be whatever you want it to be:

- assignments
- a list of discourse referents (Vermeulen)
- situations
- file cards (Heim)
- a relational database of common assumptions (Lebedeva)
- a pair of (Asher & Pogodalla)
  - available discourse references
  - modal base
Interacting with contexts

We will revise this later:

- **sel**: $\gamma \rightarrow e$
  - chooses a salient individual from the context
- **upd**: $\gamma \rightarrow e \rightarrow \gamma$
  - updates the context by introducing a discourse referent

**Notation**

$g^x$ means **upd** $g \ x$
The intuition
Everything is interpreted with respect to a context

On types

\[ \bar{e} = \gamma \rightarrow e \]
\[ \bar{t} = \gamma \rightarrow t \]
Inherent Context Sensitivity

Pronouns are inherently context sensitive

\( he := \lambda g.\text{sel } g \)

They must access the context!

Determiners are inherently context sensitive

\( det := \lambda P, Q, g.\text{det}(\lambda x.P \ K x \ g^x)(\lambda x.Q \ K x \ g^x) \)

They update the context
Context homomorphism

\[
\begin{align*}
\text{con} & \quad \text{con(he)} = \text{sel} \\
\text{con(det)} & = \lambda P, Q, g. \text{det}(\lambda x. P(\mathbb{K}x)(g^x))(\lambda x. Q(\mathbb{K}x)(g^x)) \\
\text{con}(k) & = \text{CON}(\mathbb{K}k)
\end{align*}
\]
Context Lifting

Context insensitive expressions are predictable

\[ \text{CON}_\alpha : (\gamma \to \alpha) \to \overline{\alpha} \]
\[ \text{CON}_e(a) := a \]
\[ \text{CON}_t(\phi) := \phi \]
\[ \text{CON}_{\alpha\beta}(f) := \text{CON}_\beta \circ Sf \circ \text{NOC}_\alpha \]
\[ := \lambda A.\text{CON}_\beta(\lambda g.f \ g \ (\text{NOC}_\alpha A \ g)) \]
Context Lifting

Context insensitive expressions are predictable

\[ \text{Noc}_\alpha : \overline{\alpha} \rightarrow (\gamma \rightarrow \alpha) \]
\[ \text{Noc}_e(A) := A \]
\[ \text{Noc}_t(\Phi) := \Phi \]
\[ \text{Noc}_{\alpha\beta}(F) := \mathcal{C}(\text{Noc}_\beta \circ F \circ \text{CON}_\alpha \circ K) \]
\[ := \lambda g, a. \text{Noc}_\beta(F (\text{CON}_\alpha (K a))) g \]
Examples

- $\text{CON}_{tt}(\mathbb{K}\text{not}) := \lambda \Phi, g. \text{not} (\Phi g)$
- $\text{CON}_{eeet}(\mathbb{K}\text{praise}) := \lambda A, B, g. \text{praise} (A g) (B g)$
- $\text{CON}_{(et)(et)t}(\mathbb{K}\text{every}) :=$
  $\lambda P, Q, g. \text{every}(\lambda x. P \mathbb{K}x g)(\lambda x. Q \mathbb{K}x g)$
An example

Start with:

\[
\begin{align*}
\text{every} & : (et)(et)t & \quad \text{’s} & : (eet)ee \\
\text{boy} & : et & \quad \text{mother} & : eet \\
\text{kiss} & : eet & \quad \text{he} & : e
\end{align*}
\]

Then

\[
\phi = \text{every}(\text{boy})(\lambda x.\text{kiss(’s(\text{mother})(\text{he}))}(x)) : t
\]

And

\[
\text{con}(\phi) : \gamma \rightarrow t
\]

Where

\[
\text{con}(\phi) \equiv \lambda g.\text{every}(\text{boy})(\lambda x.\text{kiss(’s(\text{mother})(sel(g^x)))})
\]
Lebedeva

a context is a logical theory (representing the commitments of the discourse participants)

\[
\gamma := t
\]

\[
\text{sel} : (e \to t) \to \gamma \to e
\]

\[
\text{sel}(P)(g) : \{ e : E | g \models P \ e \}
\]

• select an individual the context can prove has property \( P \)

\[
\text{upd} : \gamma \to t \to \gamma
\]

\[
\text{upd } g \phi := \phi \land g
\]
I bought a car, but the rear-view-mirror was broken

**World knowledge**  cars have rear-view-mirrors

\[
\llbracket \text{the} \rrbracket (\llbracket \text{r-v-m} \rrbracket) (\llbracket \text{car} \rrbracket (u) \land g)
\]

\[
g \cup \text{wk} \models \exists y. \text{r-v-m}(y) \land \text{has}(u)(y)
\]
Every boy loves his mother, but every girl hates her.

World knowledge humans have mothers

Frame problem We know too much stuff

Every boy loves his mother,...

• ‘mother’ world knowledge becomes salient (Wilks ’73; Hobbes ’78)
• a semantic version of the full NP analysis
Summary

• can study logic of context-sensitivity
  • independently of dynamics
Putting it all together

Examples
Laughing jumping boys

Start with:

\[ a : (et)(et)t \quad \text{boy} : et \]
\[ \text{jump} : et \quad \text{laugh} : et \]
\[ \text{and} : ttt \quad \text{he} : e \]

Let

\[ \phi = \text{and}(a(\text{boy})(\text{jump}))(\text{laugh}(\text{he})) : t \]

Then

\[ \text{con}(\text{dyn}(\phi)) : (g \to t) \to g \to t \]

Where

\[ \text{con}(\text{dyn}(\phi)) \equiv \lambda \psi, g. a(\text{boy})(\lambda x. \text{boy } x \land \text{laugh} (\text{sel } g^x) \land \psi g^x) \]
Definition donkey_imp :=
((some o> farmer) o> (own <o+ (some o> donkey))) implies (he o> (beat <o+ it)).

Example meanS_donkey_imp_ex :
(denotation donkey_imp (fun _ => True) True)
<->
(forall x y : E,
   Farmer x /
   Donkey y /
   Own y x ->
   (Beat
    (Select (fun x => ~ Human x)
     (Donkey y /
      Farmer x))
    (Select Human
     (Donkey y /
      Farmer x))))).
Definition donkey_rel :=
  (every o> (farmer <o (who o> (own <o+ (some o> donkey)))))) o> beat <o+ it.

Example meanS_donkey_rel_ex :
  (denotation donkey_rel (fun _ => True) True)

<->
  forall x : E,
  (exists y : E, Donkey y \ Own y x
   \ Farmer x) ->
  exists y : E,
  Donkey y \ Own y x \ Farmer x \ (Beat (Select (fun x => ~ Human x)
  (Donkey y \
   (exists z : E,
    Donkey z \ Own z x \ Farmer x)))
   x).
Summary
Dynamic Semantics

Dynamic semantics is the combination of two domains:

• context-sensitivity
• discourse semantics

On types

\[ \bar{e} = \gamma \rightarrow e \]
\[ \bar{t} = (\gamma \rightarrow t) \rightarrow \gamma \rightarrow t \]
\[ \equiv \gamma \rightarrow (\gamma \rightarrow t) \rightarrow t \]

The intuition

• individuals are interpreted wrt a context
• discourses are functions from contexts to propositions
• a sentence takes a context, and a discourse, and interprets that discourse in an updated context
Conclusion

- Modular approach allows us to
  - understand logic and structure of phenomena *an sich*
  - and view ‘the world’ as the interaction/interleaving of simpler parts
    - with proof that lifting preserves semantics
- Granny’s theory of pronouns very workable
  - modular
    - delimits role of semantics and role of ‘inference’
  - extensible
  - intuitive
Dynamics
Dynamic Montague Grammar

On types

\[
\bar{e} = \gamma \rightarrow e \\
\bar{t} = \gamma \rightarrow \gamma \rightarrow t
\]

The intuition

- Individuals are interpreted in a context
- Sentences have context-change potential
Dynamic Lifting

\[ \text{Dyn}_\alpha : (\gamma \rightarrow \alpha) \rightarrow \overline{\alpha} \]

\[ \text{Dyn}_e(a) := a \]

\[ \text{Dyn}_t(\phi) := \lambda g, h. h = h \land \phi g \]

\[ \text{Dyn}_{\alpha\beta}(f) := \lambda A. \text{Dyn}_\beta(\lambda g.f \ g \ (\text{Sta}_\alpha A \ g)) \]
Dynamic Lifting

\[ \text{STA}_\alpha : \overline{\alpha} \to \gamma \to \alpha \]

\[ \text{STA}_e (A) := A \]

\[ \text{STA}_t (\Phi) := \lambda g. \exists h. \Phi \ g \ h \]

\[ \text{STA}_{\alpha\beta} (F) := \lambda g, a. \text{STA}_\beta (F (\text{DYN}_\alpha a)) \ g \]
Examples

• $\text{DYN}_{tt}(\text{id}) := \lambda \Phi, g, h.g = h \land \exists k. \Phi g k$

• $\text{DYN}_{tt}(\text{not}) := \lambda \Phi, g, h.g = h \land \text{not} (\exists k. \Phi g k)$

• $\text{DYN}_{ttt}(\text{or}) := \lambda \Phi, \Psi, g, h.g = h \land \exists k. \Phi g k \land \Psi g k$
• $\text{dyn}(\text{and}) := \lambda \Phi, \Psi, g, h. \exists k. \Phi g k \land \Psi k h$

• $\text{dyn}(\exists) := \lambda P, g, h. \exists (\lambda x. P \ \Box x \ g^x \ h)$
Differences

• Truly *dynamic*
• Reification of contexts (cf. Quine on existing)
• no (obvious) decomposition of dynamics and context-sensitivity possible