Topics: demonstrate basic principles and tools in formal approaches to natural language meaning

Relevance: linguists, logicians, computer scientists, and philosophers

Prior knowledge:
- sets, relations, functions
- aims of theoretical linguistics

Reading: Edinburgh University Press, April 2016

www.phil.uu.nl/~yoad/efs/main.html
1. **Formal semantics**: the study of logical meaning in natural language

2. **Types and meaning composition**: How to describe properties of meanings? How does language allow us to combine them?

3. **Tutorial 1** (Sunday, 9:30-10:00)

4. **Generalized quantifiers**: How to describe meanings of expressions involving *quantity*, like *all*, *some* and *most*?

5. **Tutorial 2** (Sunday, 2:00-2:30)

6. **Spatial expressions**: Meanings of relations in *space*, such as *above*, *in* and *between*.

7. **Abstract categorial grammar / ExpSem**: long distance dependencies / experiments reciprocals
Exercises

Will be given before tutorial sessions.

It is recommended to try to solve them before the tutorial.
Soundbites

Intersective vs. non-intersective adjectives:
Tina is a *Chinese* pianist and a biologist
↔ Tina is a pianist and a *Chinese* biologist
Tina is a *skillful* pianist and a biologist
↔/→ Tina is a pianist and a *skillful* biologist

Monotonicity and Negative Polarity Items:
If John ever goes to Moscow he will have fun
*If John goes to Moscow he will ever have fun*

Spatial reasoning:
Dan is close to a gas station
↔→ *Dan is close to some gas station*
Dan is far from a gas station
↔→ *Dan is far from every gas station*
Session 1:
Basic Notions
Formal Semantics: 
History and Principles
Two prominent logicians on meaning

Gottlob Frege (1848-1925)  Alfred Tarski (1902-1983)

**Frege:** Meanings are *composed to each other*.

**Tarski:** Meanings can be described as objects in a *mathematical world*, external to language itself.
“It seems clear, then that undeniable, though only imperfect correspondences hold between formal and semantic features in language.”

(Syntactic Structures, 1957)

“...Chomsky would say [that] the semantic purposes do not determine the form of the syntax or even influence it in any significant way.”

(Chomsky's Revolution in Linguistics, John R. Searle, 1972)
Towards a Synthesis

"There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of language within a single natural and mathematically precise theory. On this point I differ from a number of philosophers, but agree, I believe, with Chomsky and his associates."

(Universal Grammar, 1970)
Frege’s Principle of Compositionality
The meaning of a compound expression is a function of the meanings of its parts, and the ways they combine with each other.
Ambiguous Expressions

I saw the man with the telescope
I saw the man with the telescope

Syntactic Ambiguity
I saw the man with the telescope
I saw the man with the telescope
Syntactic-Semantic Ambiguity

I saw the man with the telescope.
Notions in Set Theory
Notions in Set Theory (1)

Description of sets and their members:

Explicit: \{1,2,3,4\}

Implicit: \{x \text{ is a natural number} : x \text{ is between 1 and 4}\}
         = \{ x \in \mathbb{N} : 1 \leq x \leq 4 \}

\{x \text{ is a natural number} : x \text{ is bigger than 90}\}
         = \{ x \in \mathbb{N} : 90 \leq x \}
Notions in Set Theory (2)

- **Element of:** \( x \in A \)
- **Subset:** \( A \subseteq B \)
- **Intersection:** \( A \cap B \)
- **Union:** \( A \cup B \)
- **Set difference:** \( A - B \)
Complement: $\overline{A}$
Notions in Set Theory (4)

Functions:

But this is not a function:
Notions in Formal Semantics
Mentalist vs. Linguistic Meaning Relations

(1)  
   a. What is common to the objects that people categorize as being red?
   b. How do people react when they are addressed with the request *please pick a blue card from this pack*?
   c. What emotions are invoked by expressions like *my sweetheart, my grandmother* or *my boss*?

(2)  
   a. How do speakers identify relations between pairs of words like *red-color, dog-animal* and *chair-furniture*?
   b. What are the relations between the use of the imperative sentence *please pick a blue card from this pack* and the use of the similar sentence *please pick a card from this pack*?
   c. How are the descriptions *my grandmother* and *my only living grandmother* related to each other in language use?

(3)  
   Red is a color / ?Red is an animal

(4)  
   The color red annoys me / ?The animal red annoys me

(5)  
   Every red thing has a color / ?Every red thing has an animal
Entailment

Tina is tall and thin $\implies$ Tina is thin

p\: premise/antecedent  \quad c\: conclusion/consequent

Jeremy knows more than four aunts of mine $\implies$ Jeremy knows at least five aunts of mine.
A dog entered the room $\implies$ An animal entered the room.
John picked a blue card from this pack $\implies$ John picked a card from this pack.

Tina is a bird $\not\implies$ Tina can fly

Tina is a bird but she cannot fly \quad \textit{vis a vis}

$\#$Tina is tall and thin, but she is not thin

\textbf{Entailment} is the indefeasible relation, denoted $S_1 \implies S_2$, between a premise $S_1$ and a valid conclusion $S_2$ expressed as natural language sentences.
More entailments

Tina is tall and thin $\implies$ Tina is thin

(3) a. Tina is excited and she (Tina) is joyful or amazed $\implies$ Tina is excited and joyful or Tina is excited or amazed.

b. Tina is excited and joyful or Tina is excited or amazed $\implies$ Tina is excited and she (Tina) is joyful or amazed.

(4) a. Tina is tall, and Ms. Turner is not tall $\implies$ Tina is not Ms. Turner.

b. Tina is tall, and Tina is not Ms. Turner $\not\implies$ Ms. Turner is not tall.

(5) a. Ms. Turner is tall, and Tina is Ms. Turner or Ms. Charles $\implies$ Tina is tall or Tina is Ms. Charles.

b. Ms. Turner is tall, and Tina is Ms. Turner or Ms. Charles $\not\implies$ Tina is tall.
Let \( \exp \) be a language expression, and let \( M \) be a model. We write \([\exp]^M\) when referring to the denotation of \( \exp \) in the model \( M \).
A semantic theory $T$ is said to satisfy the **truth-conditionality criterion** (TCC) if for all sentences $S_1$ and $S_2$, the following two conditions are equivalent:

I. Sentence $S_1$ intuitively entails sentence $S_2$.

II. For all models $M$ in $T$: $\text{[[} S_1 \text{]]}^M \leq \text{[[} S_2 \text{]]}^M$.
Assumptions about our models

Every model has a set $E$ of **entities**.
In every model, $[[\text{Tina}]]$ is an entity.
In every model, $[[\text{tall}]]$ and $[[\text{thin}]]$ are sets of entities.

\[
\text{is}(x, A) = \begin{cases} 
1 & \text{if } x \in A \\
0 & \text{if } x \notin A 
\end{cases}
\]

$\text{AND}(A, B) = A \cap B$ = the set of all members of $E$ that are both in $A$ and in $B$

**Thus:**

\[
[[\text{Tina is thin}]]^M = \text{is}(\text{tina}, \text{thin})
\]

\[
[[\text{Tina is tall and thin}]]^M = \text{is}(\text{tina}, \text{AND}(\text{tall}, \text{thin}))
\]

**Convention:**

*Let blik be a word in a language. When the denotation $[[\text{blik}]]^M$ of blik is arbitrary, we mark it **blik**, and when it is constant across models we mark it **BLIK**. In both notations the model $M$ is implicit.*
In all three models we have:

\[
is(tina, \text{AND}(tall, thin)) \leq is(tina, thin)\]
All pianists are composers, and Tina is a pianist.
All composers are pianists, and Tina is a pianist.

**Compositionality**: The denotation of a complex expression is determined by the denotations of its immediate parts and the ways they combine with each other.
Structural ambiguity (1)

Tina is not tall and thin.
Tina is not tall, and thin.

\[ \text{AP} \rightarrow \text{tall, thin, ...} \]
\[ \text{AP} \rightarrow \text{AP and AP} \]
\[ \text{AP} \rightarrow \text{not AP} \]

A.

```
S
  / \  \
Tina is AP
  /   \  \
AP and thin
  /  \  \
not tall
```

B.

```
S
  /  \  \
Tina is AP
    /  \  \
  not tall
    /   \  \
     tall and thin
```
**Structural ambiguity (2)**

\[
\neg(A) = \overline{A} = E \setminus A = \text{the set of all the members of } E \text{ that are not in } A
\]

A.

\[
\text{IS}(\text{tina, AND(\neg(tall), thin)})
\]

\[
\text{tina, IS, AND(\neg(tall), thin)}
\]

\[
\neg(tall) \text{ AND } \text{thin}
\]

\[
\neg \text{tall}
\]

B.

\[
\text{IS}(\text{tina, NOT(AND(tall, thin))})
\]

\[
\text{tina, IS, NOT(AND(tall, thin))}
\]

\[
\neg \text{AND(tall, thin)}
\]

\[
\text{tall, AND, thin}
\]

\[
\text{IS}(\text{tina, AND(\neg(tall), thin))) = 1, \text{ i.e. } \text{tina} \in \overline{\text{tall}} \cap \text{thin}
\]

\[
\text{IS}(\text{tina, NOT(AND(tall, thin))) = 1, i.e. } \text{tina} \in \overline{\text{tall} \cap \text{thin}}
\]

Note: Ambiguity vs. vagueness
Summary – Main Notions

Facts:
Entailment (indefeasible)

Theory:
Model
Denotation in model

Adequacy:
Truth-Conditionality Criterion

Compositionality

Structural ambiguity
Session 2:
Types and Meaning composition
This Lecture

A general theory of model structure; a general semantic practice for meeting the TCC.

1 – working with types and denotations

2 – using lambda notation

3 – restricting denotations
Working with types and denotations
Basic/Complex Types and Domains

A **type** is a label for part of a model that is called a **domain**.

**Basic** types and domains:
- \( e : D_e \) - arbitrary - of entities
- \( t : D_t = \{0,1\} \) - of truth-values

**Complex** types and domains: defined inductively from basic types and domains.
Complex Types – Example

E = D_e = the set of entities \{t,j,m\}

[[\text{thin}]] = T = \{t,j\}

We can also define T as a function from D_e to D_t:

\begin{align*}
  t & \rightarrow 1 \\
  j & \rightarrow 1 \\
  m & \rightarrow 0
\end{align*}

This function characterizes T in E = D_e.

D_{et} of the complex type \text{et} is the domain of such functions.
## Characteristic functions over \{t, j, m\}

<table>
<thead>
<tr>
<th>Subset of $D_e$</th>
<th>Function in $D_{et}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$f_1: \ t \mapsto 0$, $j \mapsto 0$, $m \mapsto 0$</td>
</tr>
<tr>
<td>${m}$</td>
<td>$f_2: \ t \mapsto 0$, $j \mapsto 0$, $m \mapsto 1$</td>
</tr>
<tr>
<td>${j}$</td>
<td>$f_3: \ t \mapsto 0$, $j \mapsto 1$, $m \mapsto 0$</td>
</tr>
<tr>
<td>${j, m}$</td>
<td>$f_4: \ t \mapsto 0$, $j \mapsto 1$, $m \mapsto 1$</td>
</tr>
<tr>
<td>${t}$</td>
<td>$f_5: \ t \mapsto 1$, $j \mapsto 0$, $m \mapsto 0$</td>
</tr>
<tr>
<td>${t, m}$</td>
<td>$f_6: \ t \mapsto 1$, $j \mapsto 0$, $m \mapsto 1$</td>
</tr>
<tr>
<td>${t, j}$</td>
<td>$f_7: \ t \mapsto 1$, $j \mapsto 1$, $m \mapsto 0$</td>
</tr>
<tr>
<td>${t, j, m}$</td>
<td>$f_8: \ t \mapsto 1$, $j \mapsto 1$, $m \mapsto 1$</td>
</tr>
</tbody>
</table>

**Table 2.1:** Subsets of $D_e$ and their characteristic functions in $D_{et}$
Let $X$ be any set.

Every subset $A \subseteq X$ gives us a function $f_A : X \to \{0, 1\}$ called the characteristic function of $A$.

It is defined as follows:

$$f_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

**Example:** $X = \{a, b, c, d\}, A = \{a, c\}$

Here are some examples of how this function $f_A$ works:

- $f_A(a) = 1$.
- $f_A(b) = 0$.
- $f_A(c) = 1$.
- $f_A(d) = 0$. 
Definitions: Types and domains

**Definition 1.** The set of types over the basic types $e$ and $t$ is the smallest set $\mathcal{T}$ that satisfies:

(i) $\{e, t\} \subseteq \mathcal{T}$

(ii) If $\tau$ and $\sigma$ are types in $\mathcal{T}$ then $(\tau \sigma)$ is also a type in $\mathcal{T}$.

$$
e, t,$$

$$ee, tt, et, te,$$

$$e(ee), e(tt), e(et), e(te), t(ee), t(tt), t(et), t(te),$$

$$(ee)e, (tt)e, (et)e, (te)e, (ee)t, (tt)t, (et)t, (te)t,$$

$$(ee)(ee), (ee)(tt), (ee)(et), (ee)(te), (tt)(ee), (tt)(tt), (tt)(et), (tt)(te)$$

**Definition 2.** For all types $\tau$ and $\sigma$ in $\mathcal{T}$, the domain $D_{\tau \sigma}$ of the type $(\tau \sigma)$ is the set $D^{D_{\tau}}_{\sigma}$ – the functions from $D_{\tau}$ to $D_{\sigma}$.

$D_{e(et)}$ is the set of functions from $D_e$ to $D_{et}$

- the functions from entities to $D_{et}$

- the functions from entities to the functions from $D_e$ to $D_t$

- the functions from entities to the functions from entities to truth-values.
Intransitive verbs

Tina smiled.

\( \text{smiley}_t(\text{tina}_e) \)
Function Application

From types \( e \) and \( et \), FA gives \( t \) (as we have seen above).

From types \( (e(et))(et) \) and \( e(et) \), FA gives \( et \).

Types \( (e(et))(et) \) and \( et \) cannot combine using FA: neither of these types is a prefix of the other.

\[
\text{tina}_e + \text{smile}_{et} = \text{smile}(\text{tina}) : t.
\]

In more general terms, our type-based rule of function application is given below.

**Function application with typed denotations:** applying a function \( f \) of type \( \tau \sigma \) to an object \( x \) of type \( \tau \) gives an object \( f(x) \) of type \( \sigma \). In short –

Types:
\[
(\tau \sigma) + \tau = \tau + (\tau \sigma) = \sigma
\]

Denotations:
\[
f_{\tau \sigma} + x_\tau = x_\tau + f_{\tau \sigma} = f(x) : \sigma
\]
Intransitive and Transitive verbs

Tina smiled.
Tina [praised Mary].

A. 

Tina
  praised
  Mary

B. 

praise(mary)(tina) : t
  tina : e 
  praise(mary) : et
    praise : e(et) 
    mary : e

praise_{e(et)} + mary_{e} = \text{praise(mary)} : et

praise(mary) + tina_{e} = (praise(mary))(tina) : t
  = \text{praise(mary)}(tina)
“Curried” Relations

\[ U = \{ \langle t, m \rangle, \langle m, t \rangle, \langle m, j \rangle, \langle m, m \rangle \} \]

\[ f_U : \begin{align*}
    t & \mapsto [t \mapsto 0 \quad j \mapsto 0 \quad m \mapsto 1] \\
    j & \mapsto [t \mapsto 0 \quad j \mapsto 0 \quad m \mapsto 1] \\
    m & \mapsto [t \mapsto 1 \quad j \mapsto 0 \quad m \mapsto 1] 
\end{align*} \]

- \( f_U \) maps the entity \( t \) to the function characterizing the set \( \{ m \} \).
- \( f_U \) maps the entity \( j \) to the function characterizing the same set, \( \{ m \} \).
- \( f_U \) maps the entity \( m \) to the function characterizing the set \( \{ t, m \} \).

When the function \( f_U \) is the denotation of the verb \textit{praise}, and the entities \( t, j \) and \( m \) are the denotations of the respective names, this is the situation where:

- Mary is the only one who praised Tina.
- Mary is the only one who praised John.
- Tina and Mary, but not John, praised Mary.
Currying

\[ F: (M \times W) \rightarrow [0, 1] \]

F gives any pair of man and woman \((m, w)\) a score \(F(m, w)\) indicating matching.

\[ G: M \rightarrow (W \rightarrow [0, 1]) \]

G gives any man \(m\) a function \(G(m)\) mapping any woman \(w\) to a score \((G(m))(w)\).

Thus, we can define: \((G(m))(w) = F(m, w)\)

We say that G is the Curried version of F, and that F is the deCurried version of G.
A numerical example of Currying

The function ADD sends every number \( N \) to:
   the function \( \text{ADD}(N) \) that sends every number \( M \) to:
       \( N+M \)

Thus: \( \text{(ADD}(N))\text{(M)} = N+M. \)

But:
   \( \text{ADD}(1) \) is the \textit{successor} function.
   \( \text{ADD}(10) \) adds ten to every number.
   Etc.

With two-place operators like `+` it’s impossible to define such functions directly.

\textbf{Note:} analogy between ADD(1) and praise(mary).
Solving Type Equations

\[ \text{Tina}_{e} \left[ \text{is}_{Y} \text{tall}_{et} \right] X \right]_{t} \]

\[ \begin{align*}
\text{Tina} &: e \\
X & \quad \text{is tall} \\
\text{Tina} &: e \\
\text{et} & \quad \text{is tall} \\
\text{Tina} &: e \\
\text{et} & \quad \text{is} : Y \\
\text{tall} &: et \\
\text{Tina} &: e \\
\text{et} & \quad \text{is} : (et)(et) \\
\text{tall} &: et
\end{align*} \]

\begin{align*}
\text{Eq. 1: } e + X &= t \\
\text{Sol. 1: } X &= et \\
\text{Eq. 2: } Y + et &= et \\
\text{Sol. 2: } Y &= (et)(et)
\end{align*}

**Conclusion:** type of \( is \) should be \((et)(et)\)
Non-arbitrary Denotations: IS

For every function $f$ in $D_{et}$: $\text{is}(f) = f$.

A.

```
   Tina
     \( \text{is} \)
      \( \text{tall} \)
```

B.

```
   \( (\text{is(tall)})(\text{tina}) : t \)
     \( \text{tina} : e \)
     \( \text{is(tall)} : et \)
       \( \text{is} : (et)(et) \)
        \( \text{tall} : et \)
```
What would be the type of IS with the following (infelicitous) structure?

[Tina is] tall

What denotation would we assume for IS?
Function application & constituency (2)

a. 

S
  /   
FUNNY  tall
    /    
  Tina  is

b. 

tall'(is'(tina')): t
       /     
is'(tina') : e  tall': et
          /     
tina': c  is': ce
Non-arbitrary Denotations: NOT

Tina [ is [ not tall ]]

\( \text{NOT is the } (et)(et) \text{ function sending every } et \text{ function } g \text{ to the } et \text{ function } \text{NOT}(g) \text{ that satisfies for every entity } x: \)

\[
(\text{NOT}(g))(x) = \begin{cases} 
1 & \text{if } g(x) = 0 \\
0 & \text{if } g(x) = 1 
\end{cases}
\]
Non-arbitrary Denotations: AND

For every two functions $f_A$ and $f_B$ in $D_\eta$, characterizing the subsets $A$ and $B$ of $D_\eta$: $(\text{AND}(f_A))(f_B)$ is defined as the function $f_{A \cap B}$, characterizing the intersection of $A$ and $B$.

**Explain:**
Tina is tall and thin $\implies$ Tina is thin

Types?
In General

Types of the form

\( at \) 1-place predicates \( smile \)
\( a(at) \) 2-place predicates \( praise \)
\( a(a(at)) \) 3-place predicates \( send \)
\( \ldots \) n-place predicates

\( aa \) modifiers
\( (1\text{-}place\ coordinators) \) \( is, not \)
\( a(aa) \) 2-place coordinators \( X \text{ and } Y \)
\( a(a(aa)) \) 3-place coordinators \( X, Y \text{ and } Z \)
\( \ldots \) n-place coordinators
Using Lambda Notation
IS as Identity Function

\[ \text{[[ Tina is tall ]] = 1} \quad -- \text{\textit{tina} denotes an entity in the set for \textit{tall}} \]

With types:

\[
(IS_{et}(et)(tall_{et}))(\text{tina}_{e})
\]

**Intuitively:** IS maps any set to itself.

**Formally:**

\[
IS_{et}(et)(et) = \text{The function sending every element } g \text{ of the domain } D_{et} \text{ to } g.
\]
IS in lambda notation

\[ \text{IS}(et)(et) \]

*The function sending every element \( g \) of the domain \( D_{et} \) to \( g \).*

Instead of writing “the function sending every element \( g \) of \( D_{et} \)” as in (55), we write “\( \lambda g_{et} \)”.

Instead of “to \( g \)” as in (55), we write “\( .g \)”.

**Thus:** \[ \lambda g_{et} \cdot g \]

**Summing up:** \[ \text{IS} = \lambda g_{et} \cdot g \]

The letter ‘\( \lambda \)’ tells us that it is a function.

The notation ‘\( g_{et} \)’ before the dot introduces ‘\( g \)’ as an *ad hoc* name for the argument of this function. The type \( et \) in the subscript of \( g \) tells us that this argument can be any object in the domain \( D_{et} \).

The re-occurrence of ‘\( g \)’ after the dot tells us that the function we define in (58) returns the value of its argument.
**Lambda notation**: When writing “$\lambda x_\tau. \varphi$”, where $\tau$ is a type, we mean: “the function sending every element $x$ of the domain $D_\tau$ to $\varphi$”.
Function application with Lambda’s

\[(\lambda g_{et}.g)(\text{tall}_{et})\]
\[= \text{tall}\]

Another example:

\[\text{succ}(x) = x + 1\]

\[\text{succ}(22) = 22 + 1\]

\[(\lambda x_\tau. x + 1)(22) = 22 + 1\]

**Function application with lambda terms:** The result \((\lambda x_\tau. \varphi)(a_\tau)\) of applying a function described by a lambda term \(\lambda x_\tau. \varphi\) to an argument \(a_\tau\), is equal to the value of the expression \(\varphi\), with all occurrences of \(x\) replaced by \(a\).
Reflexives in object position

Tina praised herself

\[ \text{praise}_{e(et)}(tina_e)(tina) \]

[[ herself ]] = tina

\[ (\text{HERSELF}_{e(et)})(et)(\text{praise}_{e(et)}))(tina_e) \]

= praise(tina)(tina)

Generalizing:

For all functions \( R \) of the domain \( D_{e(et)} \), for all entities \( x \) of the domain \( D_e \):

\[ (\text{HERSELF}_{e(et)})(et)(R))(x) = R(x)(x) \]
Reflexives in object position (cont.)

For all functions \( R \) of the domain \( D_{e(\text{et})} \), for all entities \( x \) of the domain \( D_e \):
\[
(\text{HERSELF}_{e(\text{et})}(\text{et}))(R)(x) = R(x)(x)
\]

\( \text{HERSELF}_{e(\text{et})}(\text{et}) \) is
the function sending every element \( R \) of the domain \( D_{e(\text{et})} \) to the function sending every element \( x \) of the domain \( D_e \) to \( R(x)(x) \).

\[
\text{HERSELF}_{e(\text{et})}(\text{et}) = \lambda R_{e(\text{et})}. \text{the function sending every element } x \text{ of the domain } D_e \text{ to } R(x)(x)
\]

\[
\text{HERSELF}_{e(\text{et})}(\text{et}) = \lambda R_{e(\text{et})}. (\lambda x_e. R(x)(x)) = \lambda R_{e(\text{et})}. \lambda x_e. R(x)(x)
\]
Verifying the derivation

\[
\text{HERSELF}_{e(et)}(et)(\text{praise}_e(et))(\text{tina}_e)
= (((\lambda R_{e(et)} \cdot \lambda x_e. R(x)(x))(\text{praise}))(\text{tina})
= (\lambda x_e. \text{praise}(x)(x))(\text{tina})
= \text{praise}(\text{tina})(\text{tina})
\]

- compositional analysis of structure (63)
- definition (70) of HERSELF
- applying HERSELF to the argument praise
- applying (HERSELF(praise)) to the argument tina
What have we learnt here?

- A useful notation for functions
- A useful rule for simplifying notation under function application
Restricting Denotations
Expressing NOT in lambda's

Tina [ is [ not tall ]]

\text{NOT} is the \((et)(et)\) function sending every \(et\) function \(g\) to the \(et\) function \(\text{not}(g)\) that satisfies for every entity \(x\):

\[
(\text{not}(g))(x) = \begin{cases} 
1 & \text{if } g(x) = 0 \\
0 & \text{if } g(x) = 1 
\end{cases}
\]

\[
\sim = \lambda x_t. 1 - x
\]

\[
\text{NOT} = \lambda g_{et}. \lambda x_e. \sim(g(x))
\]

\[
(\text{IS}_{(et)(et)}(\text{NOT}_{(et)(et)}(\text{tall}_{et}))(\text{tina}_e))
= ((\lambda g_{et}. g)(\text{NOT}(\text{tall}))(\text{tina}))
= (\text{NOT}(\text{tall}))(\text{tina})
= ((\lambda g_{et}. \lambda x_e. \sim(g(x)))(\text{tall}))(\text{tina})
= ((\lambda x. \sim(\text{tall}(x))))(\text{tina})
= \sim(\text{tall}(\text{tina}))
\]

- comopositional analysis of structure (37)
- definition of \text{is} as identity function
- applying identity function to \text{not}(\text{tall})
- definition (55) of \text{not}
- applying definition of \text{not} to \text{tall}
- application to \text{tina}
Expressing ANDs in lambda's

\[[ \text{Tina [ is tall ]] [ and [ Tina [ is thin ]]]]\]

For any two truth-values \(x\) and \(y\): the truth-value \(x \land y\) is \(x \cdot y\), the multiplication of \(x\) by \(y\).

\[\text{AND}^t = \lambda x_t. \lambda y_t. y \land x\]

\[[ \text{Tina [ is [ tall [ and thin ]]]}\]

For every two functions \(f_A\) and \(f_B\) in \(D_{et}\), characterizing the subsets \(A\) and \(B\) of \(D_e\): 
\((\text{AND}(f_A))(f_B)\) is defined as the function \(f_{A \cap B}\), characterizing the intersection of \(A\) and \(B\).

\[\text{AND}^{et} = \lambda f_{et}. \lambda g_{et}. \lambda x_e. g(x) \land f(x)\]
Tina is a *tall* woman; the *tall* engineer visited us; I met five *tall* astronomers.

Tina is a Chinese pianist $\iff$ Tina is Chinese and Tina is a pianist.

My doctor has a white Volkswagen $\iff$ My doctor’s Volkswagen is white.

Mary saw three carnivorous animals $\iff$ Three animals that Mary saw are carnivorous.

\[
\text{Tina} \ [ \text{is } \text{ a pianist } ]
\]

\[
A(et)(et) = IS = \lambda g \text{et}.g
\]

\[
(IS(A(\text{pianist}))(tina)) = \text{pianist}(\text{tina})
\]
Attributive adjectives (2) - Intersective

Tina [ is [ a [ Chinese pianist ]]]

\[
\text{chinese}^{\text{mod}}_{(et)(et)} = \lambda f_{et}. \lambda x_e. \text{chinese}(x) \land f(x)
\]

For any two truth-values \( x \) and \( y \): the truth-value \( x \land y \) is \( x \cdot y \), the multiplication of \( x \) by \( y \).

\[
\begin{align*}
\text{is}(A(\text{chinese}^{\text{mod}}(\text{pianist}))(\text{tina})) &= (\text{chinese}^{\text{mod}}(\text{pianist}))(\text{tina}) \\
&= ((\lambda f_{et}. \lambda x_e. \text{chinese}(x) \land f(x))(\text{pianist}))(\text{tina}) \\
&= (\lambda x_e. \text{chinese}(x) \land \text{pianist}(x))(\text{tina}) \\
&= \text{chinese}(\text{tina}) \land \text{pianist}(\text{tina})
\end{align*}
\]

- composition analysis of (73)
- applying \text{is} \text{ and} \text{ A} (identity functions)
- definition (74) of \text{chinese}^{\text{mod}}
- applying modificational denotation to \text{pianist}
- applying result to \text{tina}

**Conclusion:** with adjectives like *Chinese* the attributive \((et)(et)\) denotation can be systematically derived from the predicative \text{et} denotation.

**Note:** this is not the case with all adjectives (cf. *skillful*).
Jan is a Chinese surgeon & Jan is a violinist
  ➔ Jan is a Chinese violinist
Jan is a skillful surgeon & Jan is a violinist
  ➔ Jan is a skillful violinist

**Conclusion 1:** skillful is not intersective.

However, skillful has a weaker property, which we call restrictivity.

Jan is a skillful surgeon
  ➔ Jan is a surgeon
Formally: \textit{M is subsective (or “restrictive”) if for every set of entities A, } M(A) \subseteq A. \\

**Conclusion 2:** \textit{skillful} is subsective. \\

In Lambdas: \\
\texttt{skillful}_{(et)(et)} = \lambda A. \lambda y. (\texttt{skillful1}_{(et)(et)} (A))(y) \land A(y)
Summary – restrictions on denotations

Constant, Combinatorial:
   IS, A, HERSELF

Constant, Logical:
   ANDs, NOTs

Arbitrary:
   tina, smile, praise, pianist, chinese (predicative use)

Logical operator on arbitrary:
   chinese$^{mod}$, skillful$^{mod}$ (attributive use)

Further: bachelor ➔ unmarried …
More on the classification of adjectives
I - Subsective functions

Intersective functions

\[ M(A) = X \cap A \]

Subsective functions

\[ M(A) \subseteq A \]

Note: any intersective function is subsective
II - Non-Subsective functions

\[ M(A) \subseteq A \]
Non-subsective adjectives

Jan is an alleged surgeon

\[ \Rightarrow \] Jan is a surgeon

**Conclusion:** *alleged* is not subsective.

**More examples (Partee):**

potential, alleged, arguable, likely, predicted, putative, questionable, disputed.
III - Non-subsective but co-subsective

Non-subsective (general)

Co-subsective

M(A) ⊈ A

M(A) ⊆ E-A

Note: any (non-trivial) co-subsective function is non-subsective
Co-subsective adjectives

This is a false diamond
\[\Rightarrow\] This is a diamond

**Conclusion 1:** false is not subsective.

**However:**
This is a false diamond
\[\Rightarrow\] This is not a diamond

**Conclusion 2:** false is co-subsective.

Formally: \( M \) is **co-subsective** (or “privative”) if for every set of entities \( A \), \( M(A) \subseteq E-A \).
non-subsective and privative: *wouldbe, past, spurious, imaginary, fictitious, fabricated* (in one sense), *mythical* (maybe debatable); there are prefixes with this property too, like *ex, pseudo, non*.

**Note:**

*John is an alleged criminal, and indeed he is a criminal.*

**Conclusion:** *alleged* is not co-subsective.
Adjectives - summary

These properties can be generalized (and studied) for modifiers of other types besides \((et)(et)\).
Note on (non)extensionality (1)

Jan is a skillful surgeon & Jan is a violinist
→ Jan is a skillful violinist

Conclusions we had: skillful is not intersective (but subsective).
Adjectives like skillful lack another property, which we extensionality.

Jan is a Dutch surgeon &
the surgeons are the same as the lawyers
→ Jan is a Dutch lawyer

Informally: An adjective ADJ is called extensional if for every two nouns N1 and N2 that denote the same set of entities, we have: \([ [ \text{ADJ N1} ] ] = [ [ \text{ADJ N2} ] ]\).

The adjective Dutch is extensional, but skillful is not:

Jan is a skillful surgeon &
the surgeons are the same as the lawyers
\(\models\) Jan is a skillful lawyer
Note on (non)extensionality (2)

It has been postulated that the non-intersectivity of many adjectives is derived from their non-extensionality, but the ways the two properties are related is unclear.
Session 3:
Generalized Quantifiers
We focus on quantificational \textit{NPs}: NPs containing determiners like \textit{many}, \textit{few}, \textit{every}, \textit{some}, \textit{most} etc.
Quantifiers – main claims

In order to describe the meaning of NPs with *determiners* (*every, some, most* etc.), we should let such NPs denote sets of subsets of $E$ – type \((et)t\).

The same type is needed for describing *NP coordination* in a general way.

Montague’s hypothesis about the matching between syntactic categories and semantic types leads us to adopt a uniform type for all NPs.

Some hard syntactic questions can then be given interesting semantic answers.
Keenan’s typology of determiners (1)

**Lexical Dets**
- every, each, all, some, a, no, several, neither, most, the, both, this, my, these, John's, ten, a few, a dozen, many, few

**Cardinal Dets**
- exactly/approximately/more than/fewer than/at most/only ten, infinitely many, two dozen, between five and ten, just finitely many, an even/odd number of, a large number of

**Approximative Dets**
- approximately/about/nearly/around fifty, almost all/no, hardly any, practically no

**Definite Dets**
- the, that, this, these, my, his, John's, the ten, these ten, John's ten

**Exception Dets**
- all but ten, all but at most ten, every...but John, no...but Mary,

**Bounding Dets**
- exactly ten, between five and ten, most but not all, exactly half the, (just) one...in ten, only SOME (= some but not all; upper case = contrastive stress), just the LIBERAL, only JOHN's

**Possessive Dets**
- my, John's, no student's, either John's or Mary's, neither John's nor Mary's

**Value Judgment Dets**
- too many, a few too many, (not) enough, surprisingly few, ?many, ?few

**Proportionality Dets**
- exactly half the/John's, two out of three, (not) one...in ten, less than half the/John's, a third of the/John's,
Keenan’s typology (2)

Partitive Dets
most/two/none/only some of the/John's, more of John's than of Mary's, not more than two of the ten

Negated Dets
not every, not all, not a (single), not more than ten, not more than half, not very many, not quite enough, not over a hundred, not one of John's

Conjoined Dets
at least two but not more than ten, most but not all, either fewer than ten or else more than a hundred, both John's and Mary's, at least a third and at most two thirds of the, neither fewer than ten nor more than a hundred

Adjectively Restricted Dets
John's biggest, more male than female, most male and all female, the last...John visited, the first ...to set foot on the Moon, the easiest...to clean, whatever...are in the cupboard
Function-Argument flip-flop
(an argument between FLIP and FLOP)

**FLIP:** NP: e + VP: et = S: t
(subject as argument)

**FLOP:** But can all those NPs denote entities? 😞

NP: (et)t + VP: et = S: t
(subject as function)

**FLIP:** NP denotes (et)t function?? 😞

**FLOP:** Yes! let’s do some work on it! 😊
Generalized Quantifiers - example

(1) Every man ran.

Let *every man* denote a *set of sets*: the set of subsets of $E$ that include the set of men:

(2) $\{B \subseteq E : \text{man}' \subseteq B\}$

In type-theoretical terms: *every man* denotes an $(et)t$ function. Application of this function to the VP denotation:

(3) $\text{run}' \in \{B \subseteq E : \text{man}' \subseteq B\}$
   $\iff \text{man}' \subseteq \text{run}'$
   “every member of the set man’ is a member of the set run’”

For instance, if $E = \{a, b, c, d\}$, $\text{man}' = \{a, b\}$ and $\text{run}' = \{a, b, c\}$, then:

$[\text{every man}] = \{B \subseteq E : \text{man}' \subseteq B\}$
$= \{B \subseteq \{a, b, c, d\} : \{a, b\} \subseteq B\}$
$= \{\{a, b\}, \{a, b, c\}, \{a, b, d\}, \{a, b, c, d\}\}$,
and thus $\text{run}' = \{a, b, c\} \in [\text{every man}]$. 
GQs - more examples

(4) Some man ran.

(5) \( \text{run}' \in \{B \subseteq E : \text{man}' \cap B \neq \emptyset\} \)
\[ \iff \text{man}' \cap \text{run}' \neq \emptyset \]
"there is an entity that is a member of both man' and run'"

(6) No man ran.

(7) \( \text{run}' \in \{B \subseteq E : \text{man}' \cap B = \emptyset\} \)
\[ \iff \text{man}' \cap \text{run}' = \emptyset \]

(8) exactly five men: \( \{B \subseteq E : |\text{man}' \cap B| = 5\} \)

(9) most men: \( \{B \subseteq E : |\text{man}' \cap B| > |\text{man}' \setminus B|\} \)
GQs - definition

NP: \((et)t\)  +  VP: et = S:t

\((et)t\) functions  \(\sim\)  sets of sets of sets of entities

**Terminology:** Any set \(Q \subseteq \wp(E)\) (a set of subsets of \(E\)) is called a *generalized quantifier* (GQ) over \(E\).
GQ Monotonicity

(10)  a. Some man ran quickly ⇒ Some man ran  
     b. Some man ran quickly ⊨ Some man ran

(11)  a. No man ran quickly ⇔ No man ran  
     b. No man ran quickly ⇐ No man ran

(12)  a. Exactly five men ran quickly ⇔ Exactly five men ran  
     b. Exactly five men ran quickly ⇐ Exactly five men ran

Some man is called an **upward monotone** \((\text{mon}^\uparrow)\) noun phrase.  
No man is called a **downward monotone** \((\text{mon}^\downarrow)\) noun phrase.  
Exactly five men is called a **non-monotone** (neither \(\text{mon}^\uparrow\) nor \(\text{mon}^\downarrow\)) noun phrase.

**Definition 1 (quantifier monotonicity)** A generalized quantifier \(Q \in \wp(\wp(E))\) is:

1. \(\text{mon}^\uparrow\) iff for all \(A \subseteq B \subseteq E\): if \(A \in Q\) then \(B \in Q\).
2. \(\text{mon}^\downarrow\) iff for all \(A \subseteq B \subseteq E\): if \(B \in Q\) then \(A \in Q\).
# Quantifiers and monotonicity - summary

<table>
<thead>
<tr>
<th>Noun phrase</th>
<th>Generalized quantifier</th>
<th>Monotonicity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I</strong> every man</td>
<td>( { B \subseteq E : \text{man}^* \subseteq B } )</td>
<td>MON\uparrow</td>
</tr>
<tr>
<td>II some man</td>
<td>( { B \subseteq E : \text{man}^* \cap B \neq \emptyset } )</td>
<td>MON\uparrow</td>
</tr>
<tr>
<td>no man</td>
<td>( { B \subseteq E : \text{man}^* \cap B = \emptyset } )</td>
<td>MON\downarrow</td>
</tr>
<tr>
<td>exactly one man</td>
<td>( { B \subseteq E :</td>
<td>\text{man}^* \cap B</td>
</tr>
<tr>
<td>at least three men</td>
<td>( { B \subseteq E :</td>
<td>\text{man}^* \cap B</td>
</tr>
<tr>
<td>fewer than five men</td>
<td>( { B \subseteq E :</td>
<td>\text{man}^* \cap B</td>
</tr>
<tr>
<td>between six and eleven men</td>
<td>( { B \subseteq E : 6 \leq</td>
<td>\text{man}^* \cap B</td>
</tr>
<tr>
<td><strong>III</strong> at least half the men</td>
<td>( { B \subseteq E :</td>
<td>\text{man}^* \cap B</td>
</tr>
</tbody>
</table>
Quantifiers in models

**every man:**

\[
\{a, b, c, d\} \\
\{a, b, c\} \{a, b, d\} \{a, c, d\} \{b, c, d\} \\
\{a, b\} \{a, c\} \{b, c\} \{a, d\} \{b, d\} \{c, d\} \\
\{a\} \{b\} \{c\} \{d\} \\
\emptyset
\]

**some man:**

\[
\{a, b, c, d\} \\
\{a, b, c\} \{a, b, d\} \{a, c, d\} \{b, c, d\} \\
\{a, b\} \{a, c\} \{b, c\} \{a, d\} \{b, d\} \{c, d\} \\
\{a\} \{b\} \{c\} \{d\} \\
\emptyset
\]

**no man:**

\[
\{a, b, c, d\} \\
\{a, b, c\} \{a, b, d\} \{a, c, d\} \{b, c, d\} \\
\{a, b\} \{a, c\} \{b, c\} \{a, d\} \{b, d\} \{c, d\} \\
\{a\} \{b\} \{c\} \{d\} \\
\emptyset
\]
(23) Tina ran.

(24) $\text{run}' \in \{B \subseteq E : \text{tina}' \in B\}$

"the set of runners is in the set of subsets of $E$ that contain $\text{tina}'"$

$\Leftrightarrow \text{tina}' \in \text{run}'$
Linguistic fact: Coordination applies freely to proper names and other NPs alike.

(25) Mary and/or John, neither Mary nor John, every woman or every man, most women and most men, many students but few teachers, one student and five teachers, the teacher and every student etc.

The denotation of these NPs is easily derived using GQs and the boolean treatment of coordination.

(26) a. Mary and John smiled.
    smile' ∈ \{A ⊆ E : m' ∈ A\} ∩ \{A ⊆ E : j' ∈ A\}
    ⇔ m' ∈ \text{smile'} ∧ j' ∈ \text{smile'}

b. Mary smiled and John smiled.
    smile' ∈ \{A ⊆ E : m' ∈ A\} ∧ smile' ∈ \{A ⊆ E : j' ∈ A\}
    ⇔ m' ∈ \text{smile'} ∧ j' ∈ \text{smile'}
NP Coordination (2)

(27)  a. Mary or John smiled.
\[ \text{smile}' \in \{ A \subseteq E : m' \in A \} \cup \{ A \subseteq E : j' \in A \} \]
\[ \iff m' \in \text{smile}' \lor j' \in \text{smile}' \]

b. Mary smiled or John smiled.
\[ \text{smile}' \in \{ A \subseteq E : m' \in A \} \lor \text{smile}' \in \{ A \subseteq E : j' \in A \} \]
\[ \iff m' \in \text{smile}' \lor j' \in \text{smile}' \]

(28)  a. Neither Mary nor John smiled.
\[ \text{smile}' \in \{ A \subseteq E : m' \in A \} \cap \{ A \subseteq E : j' \in A \} \]
\[ \iff m' \not\in \text{smile}' \land j' \not\in \text{smile}' \]

b. Mary didn’t smile and John didn’t smile.
\[ \overline{\text{smile}'} \in \{ A \subseteq E : m' \in A \} \land \overline{\text{smile}'} \in \{ A \subseteq E : j' \in A \} \]
\[ \iff m' \not\in \text{smile}' \land j' \not\in \text{smile}' \]
Quantifiers with VP coordination

This is in agreement with the old (and ill-defined) transformational rule of conjunction reduction (CR). However, consider the following:

(29)  
  a. NP sang and danced $\not\Leftrightarrow$ NP sang and NP danced  
      $\text{NP} = \text{some man, no man, not every man, Mary or John, at least/most}$  
      $\text{five women, exactly five women, most women}$  
  b. NP sang and danced $\Leftrightarrow$ NP sang and NP danced  
      $\text{NP} = \text{every man, Mary, Mary and John}$

(30)  
  a. NP sang or danced $\not\Leftrightarrow$ NP sang or NP danced  
      $\text{NP} = \text{every man, no man, not every man, Mary and John, at least/most}$  
      $\text{five women, exactly five women, most women}$  
  b. NP sang or danced $\Leftrightarrow$ NP sang or NP danced  
      $\text{NP} = \text{some man, Mary, Mary or John}$
Against conjunction reduction

(31) Some man danced and sang.
\[ \text{dance'} \cap \text{sing'} \in \{ A \subseteq E : \text{man'} \cap A \neq \emptyset \} \]
\[ \iff \text{man'} \cap (\text{dance'} \cap \text{sing'}) \neq \emptyset \]

This can be false when both man' \cap dance' and man' \cap sing' are non-empty.

(32) Every man danced and sang.
\[ \text{dance'} \cap \text{sing'} \in \{ A \subseteq E : \text{man'} \subseteq A \} \]
\[ \iff \text{man'} \subseteq \text{dance'} \cap \text{sing'} \]
This holds iff man' \subseteq dance' and man' \subseteq sing'.

**Conclusion:** The boolean semantics of GQs is much more fine-grained than any syntactic account of the semantics of coordination.
Determiner expressions

What is \( X \)?

Determiners like the ones considered above denote functions from sets to GQs:

\[ X + et = (et)t \]

Thus, determiners denote functions of type \((et)((et)t)\).
Alternatively, we can view such functions as relations between subsets of \( E \):

\[ \text{every}'(A) = \{ B \subseteq E : A \subseteq B \} \]

\[ \text{every}'(A)(B) \text{ iff } A \subseteq B \]

\[ \text{some}'(A)(B) \text{ iff } A \cap B \neq \emptyset \]

\[ \text{most}'(A)(B) \text{ iff } |A \cap B| > |A \setminus B| \]

**Terminology:** Any set \( D \subseteq \wp(E) \times \wp(E) \) (a relation between subsets of \( E \)) is called a **determiner function** over \( E \).
Determiner Monotonicity

\[ \text{[Every [tall man]] [ran quickly]} \Rightarrow \text{[Every [tall man]] ran} \]
\[ \Rightarrow \text{[Every [tall man from Japan]] [ran quickly]} \]
\[ \Rightarrow \text{[Every [tall man from Japan]] ran} \]
\[ \Rightarrow \text{[Every [tall man from Japan who sings the blues]] [ran quickly]} \]
\[ \Rightarrow \text{[Every [tall man from Japan who sings the blues]] ran} \]

\[ \text{[Every N] [ran quickly]} \Rightarrow \text{[Every N] ran} \]

A determiner relation \( D \) over \( E \) is called \textbf{right upward monotone} \( (\text{MON}^\uparrow) \) if and only if for all \( A \subseteq E \) and \( B_1 \subseteq B_2 \subseteq E \): \( D(A, B_1) \Rightarrow D(A, B_2) \).

A determiner relation \( D \) over \( E \) is called \textbf{right downward monotone} \( (\text{MON}^\downarrow) \) if and only if for all \( A \subseteq E \) and \( B_1 \subseteq B_2 \subseteq E \): \( D(A, B_2) \Rightarrow D(A, B_1) \).
Determiner Monotonicity – left argument

[Every man] ran \Rightarrow [Every [tall man]] ran.

Suppose that the relation $\text{EVERY}(A_1, B)$ holds. Then $A_1 \subseteq B$. By our assumption $A_2 \subseteq A_1$, we also have $A_2 \subseteq B$. Thus, the relation $\text{EVERY}(A_2, B)$ holds.

Some man ran $\nRightarrow$ Some tall man ran.
Some tall man ran $\Rightarrow$ Some man ran.

Exactly one man ran $\nRightarrow$ Exactly one tall man ran.
Exactly one tall man ran $\nRightarrow$ Exactly one man ran.
## Determiners - summary

<table>
<thead>
<tr>
<th>Determiner relations</th>
<th>Monotonicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVERY $(A, B)$</td>
<td>$\Leftrightarrow A \subseteq B$</td>
</tr>
<tr>
<td>SOME $(A, B)$</td>
<td>$\Leftrightarrow A \cap B \neq \emptyset$</td>
</tr>
<tr>
<td>NO $(A, B)$</td>
<td>$\Leftrightarrow A \cap B = \emptyset$</td>
</tr>
<tr>
<td>EXACTLY_1 $(A, B)$</td>
<td>$\Leftrightarrow</td>
</tr>
<tr>
<td>AT_LEAST_3 $(A, B)$</td>
<td>$\Leftrightarrow</td>
</tr>
<tr>
<td>FEWER_THAN_5 $(A, B)$</td>
<td>$\Leftrightarrow</td>
</tr>
<tr>
<td>BETWEEN_6_AND_11 $(A, B)$</td>
<td>$\Leftrightarrow 6 \leq</td>
</tr>
<tr>
<td>AT_LEAST_HALF $(A, B)$</td>
<td>$\Leftrightarrow</td>
</tr>
</tbody>
</table>
Negative polarity items (1)

(33)  a. John hasn’t ever been to Moscow.
     b. *John has ever been to Moscow.

(34)  a. John didn’t see any birds on the tree.
     b. *John saw any birds on the tree.

(35)  a. No student here has ever been to Moscow.
     b. *Some/every student here has ever been to Moscow.

(36)  a. Neither John nor Mary saw any birds on the tree.
     b. *Either John or Mary saw any birds on the tree.

(37)  a. None of John’s students has ever been to Moscow.
     b. *One of John’s students has ever been to Moscow.

(38)  a. Not a single student here has ever been to Moscow.
     b. *A single student here has ever been to Moscow.

(39)  a. Not more than five students here have ever been to Moscow.
     b. *More than five students here have ever been to Moscow.

(40)  a. Fewer than five students here have ever been to Moscow.
     b. *More than five students here have ever been to Moscow.
Negative polarity items (2)

(41)  a. At most four students here have ever been to Moscow.
     b. *At least four students here have ever been to Moscow.

(42)  a. Less than half the students here have ever been to Moscow.
     b. *More than half the students here have ever been to Moscow.

(43)  a. Neither any students nor any teachers attended the meeting.
     b. *Either any students or any teachers attended the meeting.

(44)  a. John neither praised nor criticized any student.
     b. *John either praised or criticized any student.

(45)  a. Every/no/at most one student who has ever been to Moscow knows about the weather there.
     b. *Some/at least one student who has ever been to Moscow knows about the weather there.

(46)  If John ever goes to Moscow he will know about the weather there.

*The Ladusaw-Fauconnier Generalization:* Negative polarity items occur within arguments of monotonic decreasing functions but not within arguments of monotonic increasing functions.
Determiner Conservativity

(20) Every man ran ⇔ Every man is a man who ran

(21) Some man ran ⇔ Some man is a man who ran

(22) Exactly five men ran ⇔ Exactly five men are men who ran

... and so on (allegedly) for all determiners!

Definition 3 (conservativity) A determiner function \( D \subseteq \wp(E) \times \wp(E) \) is called conservative iff for all \( A, B \subseteq E \): \( D(A)(B) \iff D(A)(A \cap B) \).

Hypothesized universal: All natural language determiners (simple and complex) denote conservative determiner functions.
Session 4:
Spatial Semantics
The Gas Station Puzzle (Iatriduo 2003)

Michael is close to a gas station. "∃"

Michael is far from a gas station. "∀"
Analyzing "a gas station" --

Background on indefinites

Indefinites as predicates:
- Michael is a driver.
- London is a city.

Indefinites as arguments:
- A driver showed me his car.
- A city was built here.

Partee (1987):
- Predicate indefinites are type et: $[[a\,\text{driver}]] = \text{driver}$
- Argument quantifiers are type (et)t: $\lambda P_{et} \exists x.\text{driver}(x) \land P(x)$
Analyzing "far from/close to" --

Background on locatives

Locatives are two-place predicates:

Michael is far from/close to London.

\[ \text{far\_from}(m, l) \]

Locatives are applied to entities that have locations:

\[ \text{LOC}(m) = m; \quad \text{LOC}(l) = L \]

A deeper semantics as two-place locative predicates:

\[ \text{FAR\_FROM}(\text{LOC}(m), \text{LOC}(l)) \]

\[ = \text{FAR\_FROM}(m, L) \]
Basic Account

The gas station puzzle reflects the *set denotation* of indefinites, together with the part-whole properties of spatial prepositions.

(1) Michael is **close to** London ↔
    For **some** part of London \( x \), Michael is close to \( x \).

(2) Michael is **far from** London ↔
    For **every** part of London \( x \), Michael is far from \( x \).

The same principles about locatives that account for (1) and (2) will be used to account for the gas station puzzle.

“A gas station” – analyzed as an **et** predicate; distances are measured from the **union** location of the **members** of this predicate.

\[
\text{FAR\_FROM}(m, \bigcup \{ \text{LOC}(x) : x \in \text{gs} \})
\]
Background - Types of locatives

Topological locatives:
   The car is *in* the garage.
   The camp is *outside* the city.

Distal locatives:
   The car is *far from* the garage.
   The camp is *close to* the city.
   Beijing is *1,318km from* Shanghai.

Projective locatives:
   The car is *left of* the garage.
   The bird is *close to* the house.
Topological Locatives

INSIDE: \( in, \) inside (of), within

OUTSIDE: \( out \ of, \) outside (of), without

a. The visitor is inside the building.
b. The visitor is outside the building.

a. The visitor is inside \( \text{some}_3 \) part of the building.
b. The visitor is outside \( \text{every}_4 \) part of the building.

For all points \( x \) and regions \( A \):

\[
\text{INSIDE}(x, A) \iff x \in A
\]

\[
\text{OUTSIDE}(x, A) \iff x \notin A
\]
Topological Locatives

**INSIDE:** *in, inside (of), within*

**OUTSIDE:** *out of, outside (of), without*

a. The visitor is inside the building.
b. The visitor is outside the building.

a. The visitor is inside some part of the building.
b. The visitor is outside every part of the building.

**Locative indefinites:**

Every vehicle or trailer which is parked outside of a garage shall display license plates with current registration tabs. **outside every garage**

I personally find the planets that formed outside of a star system more fascinating than ejecta. **outside every star system**

One-third of the funded proposals shall serve schools within a Metropolitan County, and at least one-third shall serve schools outside of a Metropolitan County. **outside every MC**
Topological Locatives - formally

a. The school is within a Metropolitan County.
b. The school is outside of a Metropolitan County.

\[
\text{inside}(x, \cup A) \iff \exists A \in A. \text{inside}(x, A)
\]
\[
\text{outside}(x, \cup A) \iff \forall A \in A. \text{outside}(x, A)
\]

a. \text{inside}(s, \text{loc}(\text{MC}))
   \iff \text{inside}(s, \cup \{\text{loc}(x) : x \in \text{mc}\})
   \iff \exists x \in \text{mc}. \text{inside}(s, \text{loc}(x))

b. \text{outside}(s, \text{loc}(\text{MC}))
   \iff \text{outside}(s, \cup \{\text{loc}(x) : x \in \text{mc}\})
   \iff \forall x \in \text{mc}. \text{outside}(s, \text{loc}(x))
Distal Locatives

**Upward monotone, downward monotone and non-monotone distal locatives:**

\[ \text{DIST}_M↑: \text{ far from, away from, more than/at least 20km (away) from} \]
\[ \text{DIST}_M↓: \text{ close to, near (to), less than/at most 20km (away) from} \]
\[ \text{DIST}_M→: \text{ exactly 20km (away) from, between 20km and 30km (away) from} \]

a. Michael is at least 20km from London.
b. Michael is at most 20km from London.

a. Michael is at least 20km from every part of London.
b. Michael is at most 20km from some part of London.

**Locative indefinites:**

a. Michael is at least 20km from a gas station.
b. Michael is at most 20km from a gas station.

a. Michael is at least 20km from every gas station.
b. Michael is at most 20km from some gas station.
Measuring distances

Definition 2 \textit{(metric function)} Let $M$ be a non-empty set, and let $d$ be a function from the cartesian product $M \times M$ to non-negative real numbers in $\mathcal{R}$. The function $d$ is called a \textit{metric function} if it satisfies the following requirements for all elements $x, y \in M$:

\begin{align*}
    d(x, y) &= d(y, x) \quad \text{(symmetry)} \\
    d(x, y) + d(y, z) &\geq d(x, z) \quad \text{(triangle inequality)} \\
    d(x, y) &= 0 \text{ iff } x = y \quad \text{(identity of indiscernibles)}
\end{align*}

Definition 3 \textit{(distance)} For every non-empty closed region $A \subseteq M$ and a point $x \in M$ not in $A$, the \textit{distance} between $x$ and $A$ is defined by:

\[ d(x, A) = \min\{d(x, y) : y \in A\} \]

\[ \text{FAR\_FROM}(x, A) \iff d(x, A) > r \]
\[ \text{CLOSE\_TO}(x, A) \iff d(x, A) < r \]
“Far from” vs. “Close to”

$$\text{FAR\_FROM}(x, \cup \mathcal{A}) \iff \forall A \in \mathcal{A}. \text{FAR\_FROM}(x, A)$$

$$\text{CLOSE\_TO}(x, \cup \mathcal{A}) \iff \exists A \in \mathcal{A}. \text{CLOSE\_TO}(x, A)$$
Projective Locatives

The dot is left of the line.
The dot is right of the line.

Non-existential locative indefinites:

The dot is left of a circle.
The dot is right of a circle.

Experimental work with Robert Grimm, Eva Poortman and Choonkyu Lee (SALT 2014)
Projective Locatives – formally

For any region $A$ and point $x \notin A$:

\[
\text{LEFT} \of (x, A) \iff \\
\text{the shortest vector from } A \text{ to } x \text{ has a non-zero \textquoteleft left of\textquoteright component}
\]

![Diagram of a line and a region with a point marked as left of the line and left of a circle]
Summary

Topological *(inside, outside)*, distal *(far from, close to)*, and projective *(behind, above)* locatives.

For all these locatives, we see pseudo-quantificational effect w.r.t. part-whole relations:

- close to London = close to some part of London
- far from London = far from every part of London

We explain the similarity between this behavior and “strange” effects with locative indefinites:

- close to a gas station = close to some gas station
- far from a gas station = far from every gas station
Thank you!

Sela Mador-Haim

Joost Zwarts
Modified Locatives

With distal modifiers: far outside of, 10km north of, deep under
With projective modifiers: diagonally above, straight in front of, right beneath

Modified outside (topological use) – pseudo-intersective:

The hotel is far outside the city center.

$$\text{FAR\_OUTSIDE}(x, A) \iff \text{FAR\_FROM}(x, A) \land \text{OUTSIDE}(x, A)$$

Non-existential locative indefinites:

a. This scene shows Roamer ships encountering a huge, derelict alien city in space, far outside of a star system.

b. Lightning can strike up to 10 miles outside of a thunderstorm.

c. The participants were primarily Caucasian and lived at least 25 miles outside of a town of 12,500 or more people.
Modified Locatives (cont.)

Modified *outside* (topological use) – pseudo-intersective:

Fido is less than 5m outside of a doghouse

There is a doghouse $X$ such that Fido is less than 5m from $X$

and for every doghouse $Y$ Fido is outside $Y$

Hence it is not truly existential
Modified Locatives (cont.)

Projective locatives + projective modifier:

a. The bird is straight above the house.
b. The bird is diagonally above the house.

Non-intersective, using shortest vector (Z&W)

Non-existential locative indefinites:

The bird is diagonally above a cloud.
Modified Locatives (cont.)

Projective locatives + distal modifier:

Kerewan is **10km south of** the Senegalsese Border.
Kerewan is **30km north of** the Senegalsese Border.

We are interested in the length of the shortest vector(s) among the **vectors north of** the border, and of the shortest vector(s) among the **vectors south of** the border.

**Not** the shortest vectors from the border (contra Z&W).
Kerewan is **30km north of** the Senegalsese Border.

Of the vectors **pointing northbound** from the Senegalese border to Kerewan, the shortest vector is 30km.
Now with locative indefinites:
Tweety is **30m above** a cloud.

= 

Of the vectors **pointing upward** from a cloud to Tweety, the shortest vector is 30km.
An alternative account - decomposition

Michael is far from a gas station.

a. Michael is [[NEG [close to]] [a gas station]]
b. Michael is [NEG [[close to] [a gas station]]]

Problems:

1. Why decomposing *far from* and not *close to*? (cf. Heim/Büring).
2. Radical decomposition: *exactly* ➔ *at least and at most*?
3. How to modify *3m outside*? As *3m not inside*?
4. How to account for non-existential effects that are not obtained by any decomposition?

The dot is *left of* a circle.

Tweety is *30m/diagonally above* a cloud.
Remarks and Speculations

- Existentiality
- Extensionality
- Specificity
- Some vs. a
- Kind readings
- NPIs
- Collectivity
Eigenspace readings and existentiality

Existential entailment/implication?

- During a car race in the desert, Michael’s Ferrari F2002 is running out of oil. Unfortunately, the oil used by Michael’s Ferrari is extremely hard to find.

  Michael is far from a gas station that sells this type of oil.

  Leonhard is far from a proof of his conjecture.

Our suggestion: only spatial eigenspace readings are existential.
- LOC function triggers existence requirement
Eigenspace readings and extensionality

a. John is far from a school.
b. John is far from a church.

Context: *All schools are churches and all churches are schools.*

a. The world is far from a social revolution.
b. The world is far from a solution to the inequality between people.

Context: *Every social revolution is (or would be) a solution to the inequality between people, and every solution to the inequality between people is (or would be) a social revolution.*

Our suggestion: all eigenspace readings are extensional.
Eigenspace readings and specificity

We’re far from a gas station that I read about in the guide.

Our suggestion: indefinites are ambiguous between properties and E-quantifiers.
Eigenspace readings and *some*

a. John is a teacher.  
   (predication, #identity)

b. John is a teacher that I read about in the press.  
   (#predication, identity)

c. #John is some teacher.  
   (*predication, #identity)

d. John is some teacher that I read about in the press.  
   (*predication, identity)

a. A dog barks.  
   (generic, #existential)

b. A dog that I know barks.  
   (#generic, existential)

c. Some dog barks.  
   (*generic, #existential)

d. Some dog that I know barks.  
   (*generic, existential)

) a. We’re far from a gas station.  
   (universal, #existential)

b. We’re far from a gas station that I read about in the guide. (=22)  
   (#universal, existential)

c. We’re far from some gas station.  
   (*universal, #existential)

d. We’re far from some gas station that I read about in the guide.  
   (*universal, existential)

Our suggestion: *some* is only existential. There is a connection between genericity and eigenspace readings.
Eigenspace readings and kind readings

McNally:

a. There was every kind of doctor at the convention.
b. Martha has been every kind of doctor.

a. *There was every doctor at the convention.
b. *Martha has been every doctor.

a. There was a doctor at the convention.
b. Martha has been a doctor.

Michael is far from the kind of gas station that sells this type of oil.

Our suggestion: following McNally – kind nouns trigger the property analysis.
Eigenspace readings and NPI/monotonicity

a. Michael is far from any gas station.
b. ?Michael is close to any gas station.

a. This park is outside any urban area.
b. ?This park is inside any urban area.

a. My country is far from Eurasia $\Rightarrow$ My country is far from Asia
b. My country is close to Eurasia $\not\Rightarrow$ My country is close to Asia

a. My country is outside Eurasia $\Rightarrow$ My country is outside Asia
b. My country is inside Eurasia $\not\Rightarrow$ My country is inside Asia

Our suggestion: the eigenspace analysis naturally exploits the monotonicity properties of spatial Ps.

a. Michael is far from a gas station $\Rightarrow$ Michael is far from a big gas station
b. Michael is close to a gas station $\not\Rightarrow$ Michael is close to a big gas station

a. This park is outside an urban area $\Rightarrow$ This park is outside an industrial urban area
b. This park is inside an urban area $\not\Rightarrow$ This park is inside an industrial urban area
The Impure-Atom analysis of Plurals

a. The house is (exactly) 10m away from the (row of) utility poles.
b. The house is (exactly) 10m away from a utility pole.

Our suggestion: the eigenspace of a plural can be a convex hull of the union of eigenspaces.
Big Picture

Existential quantification (specificity)

Set

Distributive quantification
Assumptions

1. Indefinites in PPs are derivationally ambiguous (Partee 87).
   - Michael is far from a gas station.
     a. Michael is [[far from] [∈ [a gas station]]]
     b. Michael is [[far from] [a gas station]]

2. Two levels of analysis of locatives (Zwarts and Winter 00).
   - Michael is far from London.
     a. Syntactic-semantic: \( \text{far\_from}(m, l) \)
     b. Conceptual-semantic: \( \text{FAR\_FROM}(m, L) \)

3. **Property-Eigenspace Hypothesis**: A property’s eigenspace is the union of eigenspaces for entities in its extension.

\[
\text{FAR\_FROM}(m, GS) \quad \text{the property GS is located at}
\]
\[
\text{LOC}(GS) = \bigcup \{ \text{LOC}(x) : x \in gs \} \quad \text{the union of gas station locations}
\]
Plan

- Overview of locatives with non-existential indefinites
- Their account using the PEH
- Why existential analyses fail
- Remarks and speculations:
  - Existentiality
  - Extensionality
  - Specificity
  - *Some* vs. *a*
  - Kind readings
  - NPIs
  - Collectivity
Lexical Reciprocity as a Typicality Preference: Experimental Evidence

Yoad Winter

Joint work with Imke Kruitwagen and Eva Poortman

ESSLLI2016, Bolzano - Bolzen, 22-26 August 2016

To appear in NELS 2016
Reciprocal verbs

**Focus:** verbs like *hug, kiss, collide*

**Two usages:**
A and B hug
A hugs B

**Old assumption:**
Reciprocity = Symmetric Participation
A and B hug $\Leftrightarrow$ A hugs B and B hugs A

**Newer assumption:**
Reciprocity *entails* Symmetric Participation
A and B hug $\Rightarrow$ A hugs B and B hugs A

**Claim:** Neither assumption is correct. The two entries are logically independent, but related through typicality.
Hypothesis: for A&B *hug*, and with many other verbs, symmetric participation is not required.
Aim

Examine whether a substantial percentage of speakers accepts reciprocity without symmetric participation above chance level, for a substantial number of reciprocal verbs.
knuffelen – “hug”
botsen (tegen) – “collide (with)”
appen – “send WhatsApp message to (each other)”
praten (tegen) – “talk (to)”
spreken (tegen) – “speak (to)”
kletsen (tegen) – “chat (to)”
roddelen (tegen) – “gossip (to)”
vechten (tegen) – “fight (against)”

Why not “talk with” etc.?
Materials – target items

One side is **active**; the other side is (visibly) **passive**. Passive side shows **collaboration**.

**Truth-judgement task for two sentences:**
- **Collective** – *het meisje en de vrouw knuffelen*
  
  “the girl and the woman hug”

- **Binary** – *het meisje knuffelt de vrouw*
  
  “the woman hugs the girl”
Materials – more target illustrations
Materials – more target illustrations
8 target verbs
X 2 sentences (collective + binary)
= 18 target items

+ 30 fillers, of two types – to hit balance between expected true/false ratios:
1. Collective/binary sentences, in situations where they are clearly true
2. Other types of sentences, in situations where they are not clearly true/false
Procedure

- 48 Dutch speakers (female 37, age M=23)
- Trials on a screen in a pseudo-random order (Open Sesame)
- green key for “true” and a red key for “false”
Control task

Appendix – 9 control items

Only collective sentences:
“the girl and the woman hug”
“the boy and the girl talk”
More control drawings
More control drawings
## Results summary

<table>
<thead>
<tr>
<th>verb</th>
<th>col+</th>
<th>bin+</th>
<th>col+bin-</th>
<th>ctrl.col+</th>
</tr>
</thead>
<tbody>
<tr>
<td>hug</td>
<td>79%</td>
<td>31%</td>
<td>48%</td>
<td>19%</td>
</tr>
<tr>
<td>collide</td>
<td>98%</td>
<td>2%</td>
<td>96%</td>
<td>65%</td>
</tr>
<tr>
<td>appen</td>
<td>94%</td>
<td>8%</td>
<td>85%</td>
<td>44%</td>
</tr>
<tr>
<td>talk</td>
<td>46%</td>
<td>4%</td>
<td>42%</td>
<td>13%</td>
</tr>
<tr>
<td>speak</td>
<td>69%</td>
<td>13%</td>
<td>56%</td>
<td>33%</td>
</tr>
<tr>
<td>chat</td>
<td>98%</td>
<td>17%</td>
<td>81%</td>
<td>27%</td>
</tr>
<tr>
<td>gossip</td>
<td>90%</td>
<td>6%</td>
<td>83%</td>
<td>46%</td>
</tr>
<tr>
<td>fight</td>
<td>73%</td>
<td>15%</td>
<td>58%</td>
<td>23%</td>
</tr>
<tr>
<td><strong>MEAN</strong></td>
<td><strong>81%</strong></td>
<td><strong>12%</strong></td>
<td><strong>69%</strong></td>
<td><strong>34%</strong></td>
</tr>
</tbody>
</table>
### Results summary

<table>
<thead>
<tr>
<th>verb</th>
<th>col+</th>
<th>bin+</th>
<th>col+bin-</th>
<th>ctrl.col+</th>
</tr>
</thead>
<tbody>
<tr>
<td>hug</td>
<td>79%</td>
<td>31%</td>
<td>48%</td>
<td>19%</td>
</tr>
<tr>
<td>collide</td>
<td>98%</td>
<td>2%</td>
<td>96%</td>
<td>65%</td>
</tr>
<tr>
<td>appen</td>
<td>94%</td>
<td>8%</td>
<td>85%</td>
<td>44%</td>
</tr>
<tr>
<td>talk</td>
<td></td>
<td></td>
<td>42%</td>
<td>13%</td>
</tr>
<tr>
<td>Changed their mind: 24-66%, $M=40%$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>speak</td>
<td></td>
<td></td>
<td>56%</td>
<td>33%</td>
</tr>
<tr>
<td>chat</td>
<td>98%</td>
<td>17%</td>
<td>81%</td>
<td>27%</td>
</tr>
<tr>
<td>gossip</td>
<td>90%</td>
<td>6%</td>
<td>83%</td>
<td>46%</td>
</tr>
<tr>
<td>fight</td>
<td>73%</td>
<td>15%</td>
<td>58%</td>
<td>23%</td>
</tr>
<tr>
<td>MEAN</td>
<td>81%</td>
<td>12%</td>
<td>69%</td>
<td>34%</td>
</tr>
</tbody>
</table>
Pilot – video clips

knuffelen – “hug”
botsen (tegen) – “collide (with)”
appen – “send WhatsApp message to (each other)”
praten (tegen) – “talk (to)”
vechten (tegen) – “fight (against)”

After showing the film, the sentence was:
“Violet and Mark hugged/collided/apped/talked/fought”
Or: “Mark hugged/… Violet”
### Results summary

<table>
<thead>
<tr>
<th>Verb</th>
<th>Col+</th>
<th>Bin-</th>
<th>Col+Bin-</th>
<th>Ctrl.Col+</th>
</tr>
</thead>
<tbody>
<tr>
<td>hug</td>
<td>64%</td>
<td>28%</td>
<td>36%</td>
<td>24%</td>
</tr>
<tr>
<td>collide</td>
<td>92%</td>
<td>0%</td>
<td>92%</td>
<td>76%</td>
</tr>
<tr>
<td>appen</td>
<td>20%</td>
<td>0%</td>
<td>20%</td>
<td>8%</td>
</tr>
<tr>
<td>talk</td>
<td>48%</td>
<td>4%</td>
<td>48%</td>
<td>8%</td>
</tr>
<tr>
<td>fight</td>
<td>48%</td>
<td>4%</td>
<td>48%</td>
<td>8%</td>
</tr>
<tr>
<td>MEAN</td>
<td>54%</td>
<td>7%</td>
<td>49%</td>
<td>25%</td>
</tr>
</tbody>
</table>
- Symmetric participation is not required with collective verbs that are traditionally classified as “reciprocal”

- Attitude of passive side matters: collaboration positively affects collective judgement

Outline of theory:
For pseudo-reciprocal predicates P, an event e is *typical* for P proportionally to two values:

- **Participation**, e.g. number of hugs
- Evidence for **collective intentionality**

*The higher the typicality value is, the higher the chance is that the event passes the speaker threshold for “truth”.*
Acknowledgements

**Support:** NWO VICI grant Winter

**Illustrations:** Ruth Noy Shapira

**Mark:** Hilbert Dijkstra

**Violet:** Lianne Zandstra

**Ideas & judgements:** Joost Zwarts, and many other colleagues and students
Abstract Categorial Grammar

Yoad Winter

Formal Semantics of Natural Language
NASSLLI2016, Rutgers University, 9-10 July 2016
Our minimalist formal semantics

- Trees over strings
- Lexical semantic types and denotations
- Inductive interpretation of trees using function application
Our minimalist formal semantics

- Trees over strings
- Lexical semantic types and denotations
- Inductive interpretation of trees using function application

Example

A.

```
Tina is tall
```

B.

```
(IS(tall))(tina) : t

  tina : e
  IS(tall) : et

  IS : (et)(et)
  tall : et
```
Three classic problems

Quantifiers in object position

*Tina praised every student*

*What do we do with types $e(et)$ and $(et)t$?
Three classic problems

Quantifiers in object position

*Tina praised every student*

What do we do with types $e(\text{et})$ and $(\text{et})t$?

Quantifier scope

*Some teacher praised every student*

How do we derive the object wide scope reading?
Three classic problems

Quantifiers in object position
Tina praised every student
What do we do with types e(\(et\)) and (\(et\))t?

Quantifier scope
Some teacher praised every student
How do we derive the object wide scope reading?

Extraction
Some teacher that Mary praised smiled
How can we interpret constituents like Mary praised?
Our modified system

- **Hypothetical Reasoning**: a dual principle to Function Application.

- **Signs**: pairs of sounds and meanings replace strings as the items manipulated by the grammar.
Function Application and Modus Ponens

Function Application (FA) Rule

\[ \tau \sigma \tau \]
\[ \sigma \]

Interpretation

\[ A \quad B \]
\[ A(B) \]

If Mary is tall then Tina is tall, and Mary is tall.

⇒ Tina is tall.
Function Application and Modus Ponens

**Function Application (FA) Rule**

<table>
<thead>
<tr>
<th>τσ</th>
<th>τ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>σ</td>
</tr>
</tbody>
</table>

**Interpretation**

\[
\frac{A}{A(B)}
\]

**Implication Elimination (Modus Ponens)**

\[
\frac{\varphi \rightarrow \psi \quad \varphi}{\psi}
\]
Function Application and Modus Ponens

**Function Application (FA) Rule**

\[
\begin{array}{c}
\tau \sigma \\
\tau \\
\hline
\sigma
\end{array}
\]

**Interpretation**

\[
\frac{A \quad B}{A(B)}
\]

**Implication Elimination** (Modus Ponens)

\[
\begin{array}{c}
\varphi \rightarrow \psi \\
\varphi \\
\hline
\psi
\end{array}
\]

If Mary is tall then Tina is tall, and Mary is tall

\[\Rightarrow \text{Tina is tall}\]
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall

(B) Tina is taller than Mary
and Mary is tall
⇒ Tina is tall
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
    ⇒ If Mary is tall then Tina is tall

(B) Tina is taller than Mary
    and Mary is tall
    ⇒ Tina is tall

Suppose we accept entailment (A). General principles of entailment, plus a general principle of conditional reasoning – *Modus Ponens* – force us to accept (B).
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall

(B) Tina is taller than Mary
and Mary is tall
⇒ Tina is tall

Suppose we accept entailment (A). General principles of entailment, plus a general principle of conditional reasoning – Modus Ponens – force us to accept (B).

Suppose we accept entailment (B). General principles of entailment, plus a general principle of conditional reasoning – which one? – should force us to accept (A).
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall

Proving (B) using (A)
Tina is taller than Mary
If Mary is tall then Tina is tall
\[\text{A}\]
Mary is tall
Tina is tall
\[\text{MP}\]

Proving (A) using (B)
Tina is taller than Mary [Mary is tall]

1
Tina is tall
\[\text{B}\]
If Mary is tall then Tina is tall
\[\text{discharge hypothesis 1}\]
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall

(B) Tina is taller than Mary
and Mary is tall
⇒ Tina is tall
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall

(B) Tina is taller than Mary
and Mary is tall
⇒ Tina is tall

Proving (B) using (A)

\[
\begin{align*}
\text{Tina is taller than Mary} \\
\text{If Mary is tall then Tina is tall} \quad \text{(A)} \\
\text{Mary is tall} \\
\text{Tina is tall} \quad \text{MP}
\end{align*}
\]
Hypothetical Reasoning – Two Equivalent Patterns

(A) Tina is taller than Mary
⇒ If Mary is tall then Tina is tall

(B) Tina is taller than Mary
and Mary is tall
⇒ Tina is tall

Proving (B) using (A)

\[
\begin{align*}
\text{Tina is taller than Mary} \\
\text{If Mary is tall then Tina is tall} \quad \text{(A)} \\
\hline \\
\text{Mary is tall} \\
\hline \\
\text{Tina is tall} \\
\text{MP}
\end{align*}
\]

Proving (A) using (B)

\[
\begin{align*}
\text{Tina is taller than Mary} \\
\text{[Mary is tall]}^1 \quad \text{(B)} \\
\hline \\
\text{Tina is tall} \\
\text{If Mary is tall then Tina is tall} \\
\text{discharge hypothesis 1}
\end{align*}
\]
Implication Introduction

... $[\varphi]^1$

\[ \vdots \]

\[ \frac{\psi}{\varphi \rightarrow \psi} \]

discharge hypothesis 1
Implication Introduction

\[ \varphi \rightarrow \psi \]

\[ \text{discharge hypothesis 1} \]

Example

\[ \varphi_1 \rightarrow (\varphi_2 \rightarrow \psi) \]

\[ \varphi_2 \rightarrow \psi \quad [\varphi_1]^1 \quad \text{MP} \]

\[ \psi \quad \varphi_2 \quad \text{MP} \]

\[ \varphi_1 \rightarrow \psi \quad \text{discharge hypothesis 1} \]
Function Abstraction

Function Introduction

\[ \ldots [\tau]^1 \]

\[
\vdots \\
\frac{\sigma}{\tau\sigma} \text{ discharge hypothesis 1}
\]
Function Abstraction

Function Introduction

\[ \vdots \]

\[ \frac{\sigma}{\tau \sigma} \quad \text{discharge hypothesis 1} \]

Example

\[ e(et) \quad [e]^1 \]

\[ \frac{et}{et} \quad \text{APP} \]

\[ e \quad \text{APP} \]

\[ \frac{t}{et} \quad \text{discharge hypothesis 1} \]
Function Abstraction – Interpretation

Function Introduction

\[ \ldots [u : \tau]^1 \]

\[ \vdash \]

\[ \frac{Z : \sigma}{\lambda u. Z : \tau \sigma} \quad \text{discharge hypothesis 1} \]
Function Abstraction – Interpretation

Function Introduction

\[ \ldots \quad [u : \tau]^1 \]
\[ \vdash \]
\[ \frac{z : \sigma}{\lambda u.z : \tau \sigma} \text{ discharge hypothesis 1} \]

Example

\[
\text{praise} : e(et) \quad [u : e]^1
\quad \frac{\text{praise}(u) : et}{\text{praise}(u)(\text{mary}) : t} \quad \frac{\text{FA}}{\lambda u.e.\text{praise}(u)(\text{mary}) : et}
\]
\[
\frac{\text{FA}}{\text{mary} : e}
\]
\[
\text{discharge hypothesis 1}
\]
Praising Mary: an intermediate summary

The constituent *praised Mary* can be analyzed in two ways.
Praising Mary: an intermediate summary

The constituent *praised Mary* can be analyzed in two ways.

Using Application: \[ \text{praise}(\text{mary}) \]

Using Abstraction: \[ \lambda u. \text{praise}(u)(\text{mary}) \]

Application (Ajdukiewicz):
- undergeneration – object quantifiers, wide scope, extraction
- overgeneration – extraction

Application + Abstraction (Lambek-Van Benthem):
- less undergeneration
- more overgeneration
Praising Mary: an intermediate summary

The constituent *praised Mary* can be analyzed in two ways.

**Using Application:**

\[
\text{praise}(\text{mary})
\]

\[
\text{mary} \quad \text{praise}
\]

**Using Abstraction:**

\[
\lambda u. \text{praise}(u)(\text{mary})
\]

\[
\text{mary} \quad \text{praise}
\]
Praising Mary: an intermediate summary

The constituent *praised Mary* can be analyzed in two ways.

**Using Application:**

\[
\text{praise(mary)}
\]

\[
\text{mary} \quad \text{praise}
\]

**Using Abstraction:**

\[
\lambda u.\text{praise}(u)(\text{mary})
\]

\[
\text{mary} \quad \text{praise}
\]

Application (Ajdukiewicz):
undergeneration – object quantifiers, wide scope, extraction
overgeneration – extraction
Praising Mary: an intermediate summary

The constituent *praised Mary* can be analyzed in two ways.

**Using Application:**

\[
praise(mary)
\]

\[
mary \quad praise
\]

**Using Abstraction:**

\[
\lambda u.\ praise(u)(mary)
\]

\[
mary \quad praise
\]

Application (Ajdukiewicz):
undergeneration – object quantifiers, wide scope, extraction
overgeneration – extraction

Application + Abstraction (Lambek-Van Benthem):
less undergeneration
more overgeneration
Using signs

“The linguistic sign unites, not a thing and a name, but a concept and a sound-image.”
(de Saussure 1916)
“The linguistic sign unites, not a thing and a name, but a concept and a sound-image.”

(de Saussure 1916)

A linguistic sign, or in short a sign, is a pair $\langle P, C \rangle$, where $P$ stands for a perceptual representation of sensory input and $C$ stands for a conceptual representation of meaning.
Using signs

“The linguistic sign unites, not a thing and a name, but a concept and a sound-image.”
(de Saussure 1916)

A linguistic sign, or in short a sign, is a pair \( \langle P, C \rangle \), where \( P \) stands for a perceptual representation of sensory input and \( C \) stands for a conceptual representation of meaning.

Sign composition:

\[
\begin{align*}
\text{MARY (sign)} & \quad \{ \text{mary (perception)} \} \\
+ & \quad \{ \text{mary (concept)} \} \\
\text{PRAISE (sign)} & \quad \{ \text{praise (perception)} \} \\
& \quad \{ \text{praise (concept)} \}
\end{align*}
\]
Using signs

“The linguistic sign unites, not a thing and a name, but a concept and a sound-image.”
(de Saussure 1916)

A linguistic sign, or in short a sign, is a pair \( \langle P, C \rangle \), where \( P \) stands for a perceptual representation of sensory input and \( C \) stands for a conceptual representation of meaning.

Sign composition:

\[
\begin{align*}
\text{MARY (sign)} & \quad \{ \text{mary (perception)} \} \\
+ & \\
\text{PRAISE (sign)} & \quad \{ \text{praise (perception)} \} \\
\end{align*}
\]

=... (two possibilities)
Strings as perceptual units

The domain of strings $D_f = F$ satisfies:

- **Closure under concatenation.** For all strings $a, b \in F$, the concatenation $a \cdot b$ is also in $F$.

- **Neutral element for concatenation.** $F$ contains an element $\epsilon$ that satisfies for every $x \in F$: $x \cdot \epsilon = \epsilon \cdot x = x$. 

Phenotypes: $f$ is a phenotype (of strings). If $\sigma$ and $\tau$ are phenotypes then $(\sigma \tau)$ is a phenotype as well. Example: In a given model:

$$
\text{tina} \cdot f = \text{tina} \quad \text{mary} \cdot f = \text{mary} \\
\text{praise} \cdot (f \cdot f) = \lambda x . \lambda y . f . y . \cdot \text{praised} \cdot \cdot x . \text{praise} \\
(f \cdot f) (\text{tina} \cdot f) (\text{mary} \cdot f) = \text{tina} \cdot \text{praised} \cdot \text{mary} \cdot \text{praise} \\
(f \cdot f) (\text{tina} \cdot f) (\text{mary} \cdot f) = \text{mary} \cdot \text{praised} \cdot \text{tina}
$$
Strings as perceptual units

The domain of strings $D_f = F$ satisfies:

- **Closure under concatenation.** For all strings $a, b \in F$, the concatenation $a \cdot b$ is also in $F$.
- **Neutral element for concatenation.** $F$ contains an element $\epsilon$ that satisfies for every $x \in F$: $x \cdot \epsilon = \epsilon \cdot x = x$.

**Pheno-types:** $f$ is a pheno-type (of strings). If $\sigma$ and $\tau$ are pheno-types then $(\sigma \tau)$ is a pheno-type as well.
Strings as perceptual units

The domain of strings $D_f = F$ satisfies:

- **Closure under concatenation.** For all strings $a, b \in F$, the concatenation $a \cdot b$ is also in $F$.

- **Neutral element for concatenation.** $F$ contains an element $\epsilon$ that satisfies for every $x \in F$: $x \cdot \epsilon = \epsilon \cdot x = x$.

**Pheno-types:** $f$ is a pheno-type (of strings). If $\sigma$ and $\tau$ are pheno-types then $(\sigma \tau)$ is a pheno-type as well.

**Example:** In a given model –

- $tina_f = tina$
- $mary_f = mary$
- $\text{praise}_{f(f)} = \lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x$
Strings as perceptual units

The domain of strings $D_f = F$ satisfies:

- **Closure under concatenation.** For all strings $a, b \in F$, the concatenation $a \cdot b$ is also in $F$.

- **Neutral element for concatenation.** $F$ contains an element $\epsilon$ that satisfies for every $x \in F$: $x \cdot \epsilon = \epsilon \cdot x = x$.

**Pheno-types:** $f$ is a pheno-type (of strings). If $\sigma$ and $\tau$ are pheno-types then $(\sigma \tau)$ is a pheno-type as well.

**Example:** In a given model –

- $\text{tina}_f = \text{tina}$
- $\text{mary}_f = \text{mary}$
- $\text{praise}_{f(f)} = \lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x$

$$\text{praise}_{f(f)}(\text{mary}_f)(\text{tina}_f) = \text{tina} \cdot \text{praised} \cdot \text{mary}$$

$$\text{praise}_{f(f)}(\text{tina}_f)(\text{mary}_f) = \text{mary} \cdot \text{praised} \cdot \text{tina}$$
Application and Abstraction using signs

Application

⟨praise \left(f \left(f \right)\right), praise \left(e \left(e \right)\right)⟩

⟨praise \left(mary \right), praise \left(mary \right)⟩

App

In our model:

= ⟨\lambda x . \lambda y . f . f . y \cdot praised \cdot x \left(mary \right), praise \left(mary \right)⟩

Abstraction

⟨praise \left(f \left(f \right)\right), praise \left(e \left(e \right)\right)⟩

\[\left[⟨u \left(f \right), u \left(e \right)⟩\right]\]

1

⟨praise \left(u \left(f \right)\right), praise \left(u \left(e \right)\right)⟩

FA

⟨mary \left(f \right), mary \left(e \right)⟩

⟨praise \left(u \left(f \right)\left(mary \right)\right), praise \left(u \left(e \right)\left(mary \right)\right)⟩

FA

⟨\lambda u . praise \left(u \left(f \right)\left(mary \right)\right), \lambda u . praise \left(u \left(e \right)\left(mary \right)\right)⟩

discharge hypothesis 1

In our model:

= ⟨\lambda u . \left(\lambda x . \lambda y . f . x \cdot praised \cdot y \right) \left(mary \right), \lambda u . praise \left(u \left(e \right)\left(mary \right)\right)⟩

= ⟨\lambda u . \left(\lambda y . f . y \cdot praised \cdot u \right) \left(mary \right), \lambda u . praise \left(u \left(e \right)\left(mary \right)\right)⟩

= ⟨\lambda u . mary \cdot praised \cdot u \left(f \right), \lambda u . praise \left(u \left(e \right)\left(mary \right)\right)⟩
Application and Abstraction using signs

Application

\[
\langle \text{praise}_{f(f)}, \text{praise}_{e(e)} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \\
\langle \text{praise}(\text{mary}), \text{praise}(\text{mary}) \rangle
\]

\[APP\]
Application and Abstraction using signs

Application

\[
\langle \text{praise}_{f( ff )}, \text{praise}_{e( et )} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \frac{\langle \text{praise}( \text{mary} ), \text{praise}( \text{mary} ) \rangle}{\langle \text{praise}( \text{mary} ), \text{praise}( \text{mary} ) \rangle} \quad \text{APP}
\]

In our model:

\[
= \langle (\lambda x_f.\lambda y_f. y \cdot \text{praised} \cdot x)(\text{mary}_f), \text{praise}(\text{mary}_e) \rangle
\]
Application and Abstraction using signs

Application

\[
\langle \text{praise}_{f(f)}, \text{praise}_{e(et)} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \\
\langle \text{praise}(\text{mary}), \text{praise}(\text{mary}) \rangle
\]

In our model:

\[
= \langle (\lambda x_f.\lambda y_f. y \cdot \text{praised} \cdot x)(\text{mary}_f), \text{praise}(\text{mary}_e) \rangle \\
= \langle (\lambda y_f. y \cdot \text{praised} \cdot \text{mary}_f), \text{praise}(\text{mary}_e) \rangle
\]
Application and Abstraction using signs

Application

\[
\langle \text{praise}_{f(f f)}, \text{praise}_{e(e t)} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \quad \langle \text{praise}(\text{mary}), \text{praise}(\text{mary}) \rangle
\]

In our model:

\[
= \langle (\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x)(\text{mary}_f), \text{praise}(\text{mary}_e) \rangle
\]

\[
= \langle (\lambda y_f. y \cdot \text{praised} \cdot \text{mary}_f), \text{praise}(\text{mary}_e) \rangle
\]

Abstraction

\[
\langle \text{praise}_{f(f f)}, \text{praise}_{e(e t)} \rangle \quad [\langle u_f, u_e \rangle]^1
\]

\[
\langle \text{praise}(u_f), \text{praise}(u_e) \rangle \quad \langle \text{mary}_f, \text{mary}_e \rangle
\]

\[
\langle \text{praise}(u_f)(\text{mary}), \text{praise}(u_e)(\text{mary}) \rangle
\]

\[
\langle \lambda u_f. \text{praise}(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle
\]

discharge hypothesis 1
Application and Abstraction using signs

Application

\[ \langle \text{praise}_{f(ff)}, \text{praise}_{e(et)} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \]

\[ \langle \text{praise}(\text{mary}), \text{praise}(\text{mary}) \rangle \]

\begin{align*}
\text{APP} \\
\text{In our model:} \\
= \langle (\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x)(\text{mary}_f), \text{praise}(\text{mary}_e) \rangle \\
= \langle (\lambda y_f. y \cdot \text{praised} \cdot \text{mary}_f), \text{praise}(\text{mary}_e) \rangle \\
\end{align*}

Abstraction

\[ \langle \text{praise}_{f(ff)}, \text{praise}_{e(et)} \rangle \langle u_f, u_e \rangle^1 \]

\[ \langle \text{praise}(u_f), \text{praise}(u_e) \rangle \]

\[ \langle \text{praise}(u_f)(\text{mary}), \text{praise}(u_e)(\text{mary}) \rangle \]

\[ \langle \text{praise}(u_f)(\text{mary}), \text{praise}(u_e)(\text{mary}) \rangle \]

\[ \langle \lambda u_f. \text{praise}(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]

\begin{align*}
\text{FA} \\
\text{FA} \\
\text{discharge hypothesis 1} \\
\text{In our model:} \\
= \langle \lambda u_f. (\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x)(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \\
\end{align*}
Application and Abstraction using signs

Application

\[\langle \text{praise}_{f(f)}, \text{praise}_{e(\text{et})} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \]
\[\Rightarrow \langle \text{praise}(\text{mary}), \text{praise}(\text{mary}) \rangle \]
\[\text{APP} \]

In our model:

\[= \langle (\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x)(\text{mary}_f), \text{praise}(\text{mary}_e) \rangle \]
\[= \langle (\lambda y_f. y \cdot \text{praised} \cdot \text{mary}_f), \text{praise}(\text{mary}_e) \rangle \]

Abstraction

\[\langle \text{praise}_{f(f)}, \text{praise}_{e(\text{et})} \rangle \left[\langle u_f, u_e \rangle\right]^1 \]
\[\Rightarrow \langle \text{praise}(u_f), \text{praise}(u_e) \rangle \]
\[\text{FA} \]
\[\Rightarrow \langle \text{praise}(u_f)(\text{mary}), \text{praise}(u_e)(\text{mary}) \rangle \]
\[\Rightarrow \langle \lambda u_f. \text{praise}(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]
\[\text{FA} \]

In our model:

\[= \langle \lambda u_f.(\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x)(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]
\[= \langle \lambda u_f.(\lambda y_f. y \cdot \text{praised} \cdot u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]
Application and Abstraction using signs

Application

\[ \langle \text{praise}_{f(f)}, \text{praise}_{e(\text{et})} \rangle \langle \text{mary}_f, \text{mary}_e \rangle \]
\[ \frac{\langle \text{praise} (\text{mary}), \text{praise} (\text{mary}) \rangle}{\text{APP}} \]

In our model:
\[ = \langle (\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x) (\text{mary}_f), \text{praise} (\text{mary}_e) \rangle \]
\[ = \langle (\lambda y_f. y \cdot \text{praised} \cdot \text{mary}_f), \text{praise} (\text{mary}_e) \rangle \]

Abstraction

\[ \langle \text{praise}_{f(f)}, \text{praise}_{e(\text{et})} \rangle \quad [\langle u_f, u_e \rangle]^1 \]
\[ \frac{\langle \text{praise} (u_f), \text{praise} (u_e) \rangle}{\text{FA}} \quad \frac{\langle \text{mary}_f, \text{mary}_e \rangle}{\text{FA}} \]
\[ \frac{\langle \text{praise}(u_f)(\text{mary}), \text{praise}(u_e)(\text{mary}) \rangle}{\text{FA}} \]
\[ \frac{\langle \lambda u_f. \text{praise}(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle}{\text{discharge hypothesis 1}} \]

In our model:
\[ = \langle \lambda u_f. (\lambda x_f. \lambda y_f. y \cdot \text{praised} \cdot x)(u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]
\[ = \langle \lambda u_f. (\lambda y_f. y \cdot \text{praised} \cdot u_f)(\text{mary}), \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]
\[ = \langle \lambda u_f. \text{mary} \cdot \text{praised} \cdot u_f, \lambda u_e. \text{praise}(u_e)(\text{mary}) \rangle \]
What have we achieved so far?

Two ways of combining the signs MARY and PRAISE:

- Direct application:

  \[ \text{PRAISE(MARY)} = \]
  
  \[
  \text{string} \quad \text{praised Mary} \\
  \text{denotation} \quad \{x \in E : x \text{ praised Mary} \}
  \]

No overgeneration!

Hypothesis

The Lambek-Van Benthem Calculus (Application + Abstraction) is a suitable logical apparatus for manipulating the composition of signs in natural language grammar.
Two ways of combining the signs MARY and PRAISE:

- **Direct application:**
  
  \[
  \text{PRAISE(MARY)} = \quad \text{string} \quad \text{praised Mary} \\
  \text{denotation} \quad \{ x \in E : x \text{ praised Mary} \}
  \]

- **With abstraction:**
  
  \[
  \lambda u. \text{PRAISE}(u)(\text{MARY}) = \quad \text{string} \quad \text{Mary praised} \\
  \text{denotation} \quad \{ y \in E : \text{Mary praised } y \}
  \]

No overgeneration!

Hypothesis: The Lambek-Van Benthem Calculus (Application + Abstraction) is a suitable logical apparatus for manipulating the composition of signs in natural language grammar.
What have we achieved so far?

Two ways of combining the signs MARY and PRAISE:

- **Direct application:**
  
  \[
  \text{PRAISE}(\text{MARY}) = \\
  \text{string} \quad \text{praised Mary} \\
  \text{denotation} \quad \{x \in E : x \text{ praised Mary}\}
  \]

- **With abstraction:**
  
  \[
  \lambda U. \text{PRAISE}(U)(\text{MARY}) = \\
  \text{string} \quad \text{Mary praised} \\
  \text{denotation} \quad \{y \in E : \text{Mary praised } y\}
  \]

No overgeneration!
What have we achieved so far?

Two ways of combining the signs MARY and PRAISE:

▶ *Direct application*:

\[
PRAISE(MARY) =
\begin{align*}
\text{string} & \quad \text{praised Mary} \\
\text{denotation} & \quad \{x \in E : x \text{ praised Mary}\}
\end{align*}
\]

▶ *With abstraction*:

\[
\lambda U. PRAISE(U)(MARY) =
\begin{align*}
\text{string} & \quad \text{Mary praised} \\
\text{denotation} & \quad \{y \in E : \text{Mary praised } y\}
\end{align*}
\]

No overgeneration!

**Hypothesis** The Lambek-Van Benthem Calculus (Application + Abstraction) is a suitable logical apparatus for manipulating the composition of signs in natural language grammar.
Relative clauses (1): using empty strings

Consider the string *that ran in some man that ran smiled*.

\[
\text{that} = \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon)
\]
Relative clauses (1): using empty strings

Consider the string \textit{that ran in some man that ran smiled}.

\[
\text{that} = \lambda P_{ff}.\lambda y_f. y \cdot \text{that} \cdot P(\epsilon)
\]

\[
\text{run} = \lambda u_f. u \cdot \text{ran}
\]
Relative clauses (1): using empty strings

Consider the string \textit{that ran in some man that ran smiled}.

\begin{align*}
\text{that} & = \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon) \\
\text{run} & = \lambda u_f. u \cdot \text{ran} \\
\text{that}(\text{run})
\end{align*}
Relative clauses (1): using empty strings

Consider the string *that ran in some man that ran smiled*.

\[
\begin{align*}
\text{that} &= \lambda P_{ff}.\lambda y_f. y \cdot \text{that} \cdot P(\epsilon) \\
\text{run} &= \lambda u_f. u \cdot \text{ran} \\
\text{that}(\text{run}) &= (\lambda P_{ff}.\lambda y_f. y \cdot \text{that} \cdot P(\epsilon))(\text{run})
\end{align*}
\]
Consider the string *that ran in some man that ran smiled*.

\[
\begin{align*}
\text{that} & = \lambda P_{ff} \cdot \lambda y \cdot y \cdot \text{that} \cdot P(\epsilon) \\
\text{run} & = \lambda u \cdot u \cdot \text{ran} \\
\text{that}(\text{run}) & = (\lambda P_{ff} \cdot \lambda y \cdot y \cdot \text{that} \cdot P(\epsilon))(\text{run}) \\
& = \lambda y \cdot y \cdot \text{that} \cdot \text{run}(\epsilon)
\end{align*}
\]
Relative clauses (1): using empty strings

Consider the string \textit{that ran} in \textit{some man that ran smiled}.

\begin{align*}
\text{that} & = \lambda P \cdot \lambda y. y \cdot \text{that} \cdot P(\epsilon) \\
\text{run} & = \lambda u. u \cdot \text{ran}
\end{align*}

\begin{align*}
\text{that}(\text{run})
& = (\lambda P \cdot \lambda y. y \cdot \text{that} \cdot P(\epsilon))(\text{run}) \\
& = \lambda y. y \cdot \text{that} \cdot \text{run}(\epsilon) \\
& = \lambda y. y \cdot \text{that} \cdot ((\lambda u. u \cdot \text{ran})(\epsilon))
\end{align*}
Consider the string \textit{that ran} in \textit{some man that ran smiled}.

\[
\begin{align*}
\text{that} &= \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon) \\
\text{run} &= \lambda u_f. u \cdot \text{ran} \\
\text{that}(\text{run}) &= (\lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon))(\text{run}) \\
&= \lambda y_f. y \cdot \text{that} \cdot \text{run}(\epsilon) \\
&= \lambda y_f. y \cdot \text{that} \cdot ((\lambda u_f. u \cdot \text{ran})(\epsilon)) \\
&= \lambda y_f. y \cdot \text{that} \cdot (\epsilon \cdot \text{ran})
\end{align*}
\]
Consider the string *that ran in some man* *that ran* smiled.

\[
\begin{align*}
\text{that} &= \lambda P_{ff}. \lambda y.f. y \cdot \text{that} \cdot P(\epsilon) \\
\text{run} &= \lambda u.f. u \cdot \text{ran} \\
\text{that}(\text{run}) &= (\lambda P_{ff}. \lambda y.f. y \cdot \text{that} \cdot P(\epsilon))(\text{run}) \\
&= \lambda y.f. y \cdot \text{that} \cdot \text{run}(\epsilon) \\
&= \lambda y.f. y \cdot \text{that} \cdot ((\lambda u.f. u \cdot \text{ran})(\epsilon)) \\
&= \lambda y.f. y \cdot \text{that} \cdot (\epsilon \cdot \text{ran}) \\
&= \lambda y.f. y \cdot \text{that} \cdot \text{ran}
\end{align*}
\]
Relative clauses (2): using empty strings

Consider the string *that Mary praised*.

\[
\text{that} = \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon)
\]
Relative clauses (2): using empty strings

Consider the string \textit{that Mary praised}.

\[ \text{that} = \lambda P_{ff}.\lambda y_f. y \cdot \text{that} \cdot P(\epsilon) \]
\[ \text{mp} = \lambda u_f. \text{mary} \cdot \text{praised} \cdot u \]
Relative clauses (2): using empty strings

Consider the string that Mary praised.

\[
\text{that} = \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon)
\]

\[
\text{mp} = \lambda u_f. \text{mary} \cdot \text{praised} \cdot u
\]

\[
\text{that}(\text{mp})
\]
Consider the string *that Mary praised*.

\[
\text{that} = \lambda P_{ff}. \lambda y_\cdot y \cdot \text{that} \cdot P(\epsilon)
\]

\[
\text{mp} = \lambda u_\cdot \text{mary} \cdot \text{praised} \cdot u
\]

\[
\text{that(mp)} = (\lambda P_{ff}. \lambda y_\cdot y \cdot \text{that} \cdot P(\epsilon))(\text{mp})
\]
Relative clauses (2): using empty strings

Consider the string *that Mary praised*.

that = λ\text{ff}.\lambda f. y \cdot \text{that} \cdot P(\epsilon)

mp = λu. \text{mary} \cdot \text{praised} \cdot u

that(mp) = (λ\text{ff}.\lambda y. f \cdot \text{that} \cdot P(\epsilon))(mp)
= λy. f \cdot \text{that} \cdot mp(\epsilon)
Consider the string that Mary praised.

that = \lambda P_{ff}.\lambda y_{f}. y \cdot that \cdot P(\epsilon)

mp = \lambda u_{f}. mary \cdot praised \cdot u

that(mp)
= (\lambda P_{ff}.\lambda y_{f}. y \cdot that \cdot P(\epsilon))(mp)
= \lambda y_{f}. y \cdot that \cdot mp(\epsilon)
= \lambda y_{f}. y \cdot that \cdot ((\lambda u_{f}. mary \cdot praised \cdot u)(\epsilon))
Relative clauses (2): using empty strings

Consider the string *that Mary praised*.

\[
\text{that} = \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon)
\]

\[
\text{mp} = \lambda u_f. \text{mary} \cdot \text{praised} \cdot u
\]

\[
\text{that(mp)} = ((\lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon))(\text{mp}) = \\
\lambda y_f. y \cdot \text{that} \cdot \text{mp}(\epsilon) = \\
\lambda y_f. y \cdot \text{that} \cdot ((\lambda u_f. \text{mary} \cdot \text{praised} \cdot u)(\epsilon)) = \\
\lambda y_f. y \cdot \text{that} \cdot (\text{mary} \cdot \text{praised} \cdot \epsilon)
\]
Relative clauses (2): using empty strings

Consider the string *that Mary praised*.

\[
\text{that} = \lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon)
\]

\[
\text{mp} = \lambda u_f. \text{mary} \cdot \text{praised} \cdot u
\]

\[
\text{that(mp)}
\]

\[
= (\lambda P_{ff}. \lambda y_f. y \cdot \text{that} \cdot P(\epsilon))(\text{mp})
\]

\[
= \lambda y_f. y \cdot \text{that} \cdot \text{mp}(\epsilon)
\]

\[
= \lambda y_f. y \cdot \text{that} \cdot ((\lambda u_f. \text{mary} \cdot \text{praised} \cdot u)(\epsilon))
\]

\[
= \lambda y_f. y \cdot \text{that} \cdot (\text{mary} \cdot \text{praised} \cdot \epsilon)
\]

\[
= \lambda y_f. y \cdot \text{that} \cdot \text{mary} \cdot \text{praised}
\]
Consider the noun phrase _someone_ in _someone ran_.

\[ \text{someone} = \lambda P_{ff}. P(\text{someone}) \]
Consider the noun phrase *someone* in *someone ran*.

\[
\text{someone} = \lambda P_{ff}. P(\text{someone}) \\
\text{run} = \lambda u_f. u \cdot \text{ran}
\]
Quantifiers

Consider the noun phrase someone in someone ran.

\[
someone = \lambda P_{ff}. P(someone) \\
run = \lambda u_{f}. u \cdot ran \\
someone(\text{run})
\]
Consider the noun phrase *someone* in *someone ran*.

\[
\text{someone} = \lambda P_{ff}. P(\text{someone}) \\
\text{run} = \lambda u_{f}.u \cdot \text{ran} \\
\text{someone}(\text{run}) \\
(\lambda P_{ff}. P(\text{someone}))(\lambda u_{f}.u \cdot \text{ran})
\]
Quantifiers

Consider the noun phrase *someone* in *someone ran*.

\[
someone = \lambda P_{ff}. P(someone) \\
run = \lambda u_f.u \cdot ran \\
someone(run) \\
(\lambda P_{ff}. P(someone))(\lambda u_f.u \cdot ran) \\
(\lambda u_f.u \cdot ran)(someone)
\]
Consider the noun phrase *someone* in *someone ran*.

\[
someone = \lambda P_{ff}. P(someone)
\]

\[
run = \lambda u_{f}. u \cdot ran
\]

\[
someone(run)\\(\lambda P_{ff}. P(someone))(\lambda u_{f}. u \cdot ran)\\(\lambda u_{f}. u \cdot ran)(someone)\\someone \cdot ran
\]

Question: How about *some* in *some man ran*?

Further: The notion *abstract type* (abstract category).
Relative clauses (3): full derivation

\[
\text{SOME} \left( \text{THAT} \left( \lambda U_{f,e}. \text{PRAISE}(U)(\text{MARY}) \right)(\text{TEACHER}) \right)(\text{SMILE})
\]

\[
\text{SOME} \left( \text{THAT} \left( \lambda U_{f,e}. \text{PRAISE}(U)(\text{MARY}) \right)(\text{TEACHER}) \right) \quad \text{SMILE}
\]

\[
\text{SOME} \quad \text{THAT} \left( \lambda U_{f,e}. \text{PRAISE}(U)(\text{MARY}) \right)(\text{TEACHER})
\]

\[
\text{TEACHER} \quad \text{THAT} \left( \lambda U_{f,e}. \text{PRAISE}(U)(\text{MARY}) \right)
\]

\[
\text{THAT} \quad \lambda U_{f,e}. \text{PRAISE}(U)(\text{MARY})
\]

\[
\text{MARY} \quad \text{PRAISE}
\]
Relative clauses (3): full derivation

\[ \text{some \cdot teacher \cdot that \cdot mary \cdot praised \cdot smiled} \]

\[ \text{SOME} \left( \text{that} \left( \lambda u_e. \text{praise}(u)(\text{mary}))(\text{teacher}) \right)(\text{smile}) \right) \]

\[ \text{SOME} \left( \text{that} \left( \lambda u_f, e. \text{PRAISE}(u)(\text{MARY}))(\text{TEACHER}) \right)(\text{SMILE}) \right) \]

\[ \text{SOME} \left( \text{that} \left( \lambda u_f, e. \text{PRAISE}(u)(\text{MARY}))(\text{TEACHER}) \right) \right) \]

\[ \text{SMILE} \]

\[ \text{TEACHER} \]

\[ \text{THAT} \left( \lambda u_f, e. \text{PRAISE}(u)(\text{MARY}) \right) \]

\[ \text{THAT} \]

\[ \lambda u_f, e. \text{PRAISE}(u)(\text{MARY}) \]

\[ \text{MARY} \]

\[ \text{PRAISE} \]
Quantificational object noun phrases

\[
\text{EVERY(\text{STUDENT})}(\lambda u_{f,e}.\text{PRAISE}(u)(\text{TINA}))
\]

\[
\text{TINA} \quad \lambda v_{f,e}.\text{EVERY}(\text{STUDENT})(\lambda u_{f,e}.\text{PRAISE}(u)(v))
\]

\[
\text{PRAISE} \quad \text{EVERY}(\text{STUDENT})
\]

\[
\text{EVERY} \quad \text{STUDENT}
\]
Quantificational object noun phrases

\[ tina \cdot praised \cdot every \cdot student \]

\[ \text{EVERY} (\text{student})(\lambda u_e \cdot \text{praise}(u)(tina)) \]

\[ \text{EVERY} (\text{STUDENT})(\lambda U_{f,e} \cdot \text{PRAISE}(U)(TINA)) \]

\[ \text{TINA} \quad \lambda V_{f,e} \cdot \text{EVERY} (\text{STUDENT})(\lambda U_{f,e} \cdot \text{PRAISE}(U)(V)) \]

\[ \text{PRAISE} \quad \text{EVERY} (\text{STUDENT}) \]

\[ \text{EVERY} \quad \text{STUDENT} \]
Quantifier scope (1): object narrow scope

\[
\text{SOME(TEACHER)}(\lambda v_{f,e}.\text{EVERY(STUDENT)}(\lambda u_{f,e}.\text{PRAISE}(u)(v)))
\]
Quantifier scope (1): object narrow scope

\[
\text{SOME}(\text{TEACHER})\left( \lambda v. \text{EVERY}(\text{STUDENT})\left( \lambda u. \text{praise}(u)(v) \right) \right)
\]

\[
\text{SOME}(\text{TEACHER})\left( \lambda v_{f,e}. \text{EVERY}(\text{STUDENT})\left( \lambda u_{f,e}. \text{PRAISE}(u)(v) \right) \right)
\]
Quantifier scope (2): object wide scope

\[
\text{EVERY(}\text{STUDENT}) (\lambda u_{f,e}. \text{SOME(TEACHER)} (\lambda v_{f,e}. \text{PRAISE}(u)(v)))
\]
Quantifier scope (2): object wide scope

\[
\text{EVERY (STUDENT)}(\lambda u_e.\text{SOME (teacher)}(\lambda v_e.\text{praise} (u)(v)))
\]

\[
\text{EVERY (STUDENT)}(\lambda u_{f,e}.\text{SOME (TEACHER)}(\lambda v_{f,e}.\text{PRAISE} (u)(v)))
\]
Quantifier scope (2): object wide scope

\[ \text{some} \cdot \text{teacher} \cdot \text{praised} \cdot \text{every} \cdot \text{student} \]
\[ \text{EVERY}((\text{student})((\lambda u_e \cdot \text{SOME}((\text{teacher})((\lambda v_e \cdot \text{praise}(u)(v)))))]) \]

\[ \text{EVERY}((\text{STUDENT})((\lambda u_{f,e} \cdot \text{SOME}((\text{TEACHER})((\lambda v_{f,e} \cdot \text{PRAISE}(u)(v)))))]) \]

Two parameters:
- Order of composition of signs – determines semantic scope
- Sign argument saturated – determines syntactic position
Lambek-Van Benthem Calculus – flexibility of hypothetical reasoning

Directionality is not in tecto-level syntax, but in the pheno-level objects that it manipulates

Saussurean signs – avoiding overgeneration

Implications:
- Model-theoretic phonology
- Free variables in grammar, not in meaning
- Syntax and semantics hand in hand
Further usages

By extending the framework with *possible worlds*, scope mechanisms as in ACG can also deal with *de dicto/de re* ambiguities, such as:

Mary is looking for a secretary.


