EXECUTING THE FREGEAN PROGRAMME

LECTURE 3

Moreno Mitrović

The HU Lectures on Formal Semantics

MINIMAL REVISION

- Sets
  - $x \in A$
  - $\varnothing \subseteq A$ (for any A)
  - Set-relations, Venn diagrams
- Functions (blackboard)
- Questions?

FREGE'S CONJECTURE

FREGE'S CONJECTURE

RECAP

QUICK THROWBACK TO FREGE

- Frege construed unsaturated meanings as functions.
- Functions are relations of mapping between a domain and a range. Or, functions are sets of ordered pairs.

∴ How can unsaturated meanings be analysed as sets of ordered pairs? Pairs of what? (Hm.)

Frege's conjecture: what was it?

Meaning composition is **function application**.

- Function application = an application of function. (Go figure.)
- But how? Think of what a function is ...
- If unsaturated meanings are functions, then what is their **domain** and **range**?
- To understand this, we'll take an excursus.

EXTENSION & FIRST APPLICATION

EXTENSION & FIRST APPLICATION

EXTENSION

Extension: what is it?

An extension of a sentence is its **truth-value**.

- What is a truth-value? How many values can truth have?
- Extension: what is said **extends** into the **real world** and bounces back as **either true or false.**
- Meaning is then extension! Another word we'll use instead of 'meaning': **denotation**.

(1) a. $x = x$

b. $[\![x]\!]$ = the denotation (meaning) of x

EXTENSION AND ⟦ ⟧

- Technically, ⟦ ⟧ is a **function**, more precisely, it is **an interpretation function**.
  - It takes *something* and returns *its meaning*.
  - More importantly: it takes **something linguistic** and returns **something actual**.
  - Romantically, then ⟦ ⟧ : WORDS ↦ MEANING
- ⟦ ⟧ is thus the great (one-directional!) truth-translator.
- We will be interpreting English using **extensional semantics**.

- A **sentence** can be true (1) or false (0).
- Can a proper name be true or false?
- If sentences 'extend' to truth-values, what do names 'extend' to?

(2) a. $[\![\textbf{Ann smokes}]\!] = \begin{cases} 1 & \text{iff Ann smokes} \\ 0 & \text{iff Ann does not smoke} \end{cases}$

b. $[\![\textbf{Ann}]\!]$ = ?Ann

- Names mean **things**. Smart talk: names denote individuals in the world.
- Can you see how we could now get closer to the meaning of the verb **smokes**? Try figure it out, in groups.
- (3 min. discussion)

Let D be **the set of all individuals that exist in the real world**. (D stands for **domain**.)

(Q: how would we define the set D?)

- Possible denotations:
  - Elements of D, the set of actual individuals. **names**
  - Elements of $\{0, 1\}$, the set of truth-values. **sentences**
  - Functions from D to $\{0, 1\}$. **everything else**
- Back to **smoking**: what does **smokes** denote?
- Answer: a function from D to $\{0, 1\}$
- Could we write it more formally?

- Our semantic theory will need **three components**. We already introduced one.

i. **Inventory of denotations** (3 of those.)
ii. **Lexicon** (For **terminal nodes only**; we automatically know the denotation type of words.)
iii. **Rules of composition**, a.k.a., rules for 'non-terminal nodes' (We'll need some syntax now.)

Inventory of denotations

- Elements of D, the set of actual individuals.
- Elements of $\{0,1\}$, the set of truth-values.
- Functions from D to $\{0,1\}$.

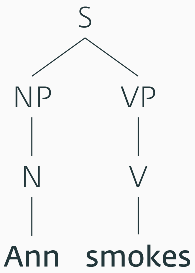
Lexicon

- Proper names:
  - $[\![$**Ann**$]\!]$ =Ann
  - $[\![$**Bill**$]\!]$ =Bill
- Intransitive verbs:
  - $[\![$**smokes**$]\!] = f : D \mapsto \{0,1\}$
    For all $x \in D, f(x) = 1$ iff x smokes
  - $[\![$**works**$]\!] = f : D \mapsto \{0,1\}$
    For all $x \in D, f(x) = 1$ iff x works

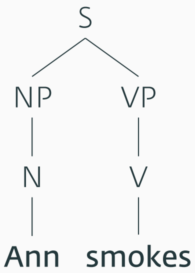
EXTENSION & FIRST APPLICATION

Some syntax

- For now, we'll be working with sentences that contain a proper name as a subject and an intransitive verb.
- Such sentences associate with a syntactic structure on the right.



- A sentence **S** comprises a subject, which is a Noun Phrase **NP**, and a verb, which is actually a Verb Phrase **VP**.
- Every phrase has a head, so **NP** contains a noun head, **N**, and the **VP** contains a verb head, **V**.
- That's the syntax we need for now.



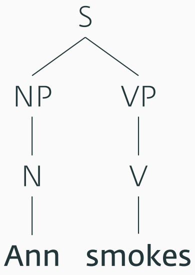
SOME SYNTAX: STRUCTURAL RELATIONS

- Let's introduce three simple **structural relations**:

motherhood α is β's **mother** (=mother node) if α immediately dominates β. List some motherhood relations.

daughterhood β is α's **daugher** (=daughter node) if α immediately dominates β. List some motherhood relations.

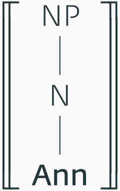
sisterhood α is β's **sister** (and vice versa) if both α and β are immediately dominated by γ. List the one sisterhood relation.

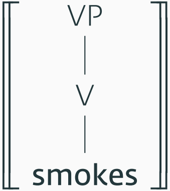


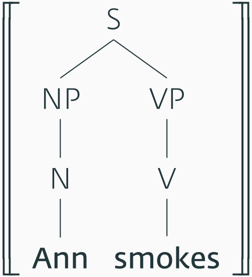
- And two more pairs of concept:

non/terminal node has (no) daughters.

non-branching node has (no) sisters.

(3) a. ⟦Ann⟧ = 

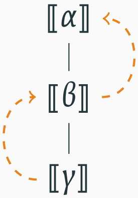
b. ⟦smokes⟧ = 

(4) ⟦Ann smokes⟧ = 

- For now, there are only two rules for interpreting trees, depending on whether the sub/tree is non/branching:

Rule #1

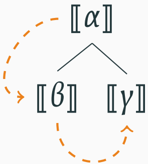
In a non-branching node, the denotation of the daughter is inherited by the mother.



$[\![\alpha]\!] = [\![\beta]\!] = [\![\gamma]\!]$

Rule #2

In a (binary) branching node, the denotation of the mother is the functional application of its daughters.



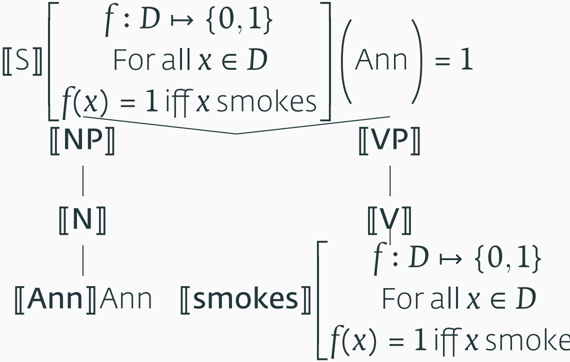
$[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$

- Let's try calculating the meaning of "**Ann smokes**" then.
- Before we do, let's recall Frege's unsaturated meanings. Is there an unsaturated meaning in "**Ann smokes**"?
- Yes. It's the verb **smokes**. If unsaturated meanings are functions, then **smokes** is a function, as we already learnt.
- What kind of a function?
- Well, it takes individuals, like **Ann**, and returns (=its values are) truth-values.
- **The extension of of an intransitive verb like "smoke", then, should be a function from individuals to truth-values.**


S
NP
VP
N
V
Ann
smokes
=
⟦S⟧
⟦NP⟧
⟦VP⟧
⟦N⟧
⟦V⟧
⟦Ann⟧
⟦smokes⟧

APPLYING THE INTERPRETATION RULES TO TREES

- **Lexicon**: denotation of ⟦**Ann**⟧
- **Lexicon**: denotation of ⟦**smokes**⟧
- **Composition rule**: non-branching nodes inherit the denotations from their daughters. This happens twice, for both the **NP** and the **VP**.
- **Composition rule**: branching nodes as FA at S-level.



Extension & first application

Back to truth-conditions

- Suppose Ann, Jan, and Maria are the only individuals in the actual world.
- Ann and Jan are the only smokers.
- The extension of the verb "smoke" can, in this world, be displayed as follows:

$$[\![\textbf{smokes}]\!] \left[\!\!\left[\begin{array}{c} \text{VP} \\ | \\ \text{V} \\ | \\ \textbf{smokes} \end{array} \right]\!\!\right] \big(\text{Ann}\big) = \begin{bmatrix} \text{Ann} \mapsto 1 \\ \text{Jan} \mapsto 1 \\ \text{Maria} \mapsto 0 \end{bmatrix} \big(\text{Ann}\big) = 1$$

EXTENSION & FIRST APPLICATION

CHARACTERISTIC FUNCTIONS

- We have construed the meaning of intransitive verbs as functions from a set of individuals to a set of truth values.
- Alternatively, the meaning of intransitive verbs can be construed simply as a **set**.
  - **Intuition**: an intransitive verb denotes the set of individuals that it is true of.

Characteristic function

a. Let A be a set. Then CHAR$_f$, the **characteristic function** of A, is that function f:

$$f(x) = \begin{cases} 1 & \text{for any } x \in A \\ 0 & \text{for any } x \notin A \end{cases}$$

b. Let f be a function with range $\{0, 1\}$. Then CHAR$_f$, **the set characterised by** f, is $\{x \in D : f(x) = 1\}$

Context

Let our universe contain only three individuals: {Ann, Jan, Maria}. Suppose that Ann and Jan are the only ones who sleep, and Ann is the only one who snores.

Example: set treatment

If intransitive verbs denote sets, then **sleep** and **snore** denote the following:

(5) a. $[\![$**sleep**$]\!]$ = {Ann, Jan}

b. $[\![$**snore**$]\!]$ = {Ann}

(6) a. Ann $\in$ $[\![$**sleep**$]\!]$

b. $[\![$**snore**$]\!]$ $\subseteq$ $[\![$**sleep**$]\!]$

Example: CHAR$_f$ treatment

Same context. If intransitive verbs denote characteristic functions (CHAR$_f$), then the following are denotations of **sleep** and **snore**.

(7) a. $[\![\textbf{sleep}]\!] = \begin{bmatrix} \text{Ann} \mapsto 1 \\ \text{Jan} \mapsto 1 \\ \text{Maria} \mapsto 0 \end{bmatrix}$

b. $[\![\textbf{snore}]\!] = \begin{bmatrix} \text{Ann} \mapsto 1 \\ \text{Jan} \mapsto 0 \\ \text{Maria} \mapsto 0 \end{bmatrix}$

(8) Are the following now true?

a. Ann $\in [\![\textbf{sleep}]\!]$ (**NO**)

b. $[\![\textbf{snore}]\!] \subseteq [\![\textbf{sleep}]\!]$ (**NO**)

- We will adopt the CHAR$_f$ notation and conception (right column) and drop basic set notation.

	Old system	**New system**
⟦V$_{\text{INTR}}$ =⟧	Set	CHAR$_f$
⟦**Ann sleeps**⟧ =	Ann ∈ ⟦**sleep**⟧	⟦**sleep**⟧(Ann) = 1
Set rel.	⟦**snore**⟧ ⊆ ⟦**sleep**⟧	$\{x :$ ⟦**snore**⟧$(x) = 1\} \subseteq$ $\{x :$ ⟦**sleep**⟧$(x) = 1\}$

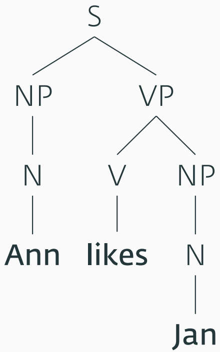
ADDING TRANSITIVE VERBS

WHAT ABOUT TRANSITIVE VERBS?

(9) Ann smokes.

(10) Ann likes Jan.

(11)



- How do we define the meaning of **likes**, given what we know about the meaning of an intransitive verb like **smokes**?

WHAT ABOUT TRANSITIVE VERBS?

(12) $[\![\text{smokes}]\!] = f : D \mapsto \{0, 1\}$

For all $x \in D$, $f(x) = 1$ iff x smokes

(13) $[\![\text{likes}]\!] = g : D \mapsto f$

(14) $[\![\text{likes}]\!] = g : D \mapsto D \mapsto \{0, 1\}$

(15) $[\![\text{likes}]\!] = f : D \mapsto \{g : g \text{ is a function from } D \mapsto \{0, 1\}\}$

For all $x, y \in D$, $f(x)(y) = 1$ iff x likes y

- This logic is in line with the syntax: V first combines with the direct object to form a VP (hence it needs a meanings).
- Recall branching-node meaning and the inventory of meanings ...(What's different here?)

Domain of individuals

e is the type of individuals (**e**ntities), where $D_{\mathsf{e}} := D$.

Domain of truth-values

t is the type of **t**ruth values, where $D_{\mathsf{t}} := \{0, 1\}$.

- **e** and **t** are basic types and correspond to Frege's **saturated meanings**.
- What, then, are **unsaturated meanings**?

- They are of derived types for various functions.

Domains of derived types

a. $D_{\langle e,t\rangle} := \{f : f \text{ is a function from } D_e \mapsto D_t\}$

b. $D_{\langle e,\langle e,t\rangle\rangle} := \{f : f \text{ is a function from } D_e \mapsto D_{\langle e,t\rangle}\}$

c. ...

Semantic types

a. **e** and **t** are semantic types.

b. If σ and τ are semantic types, then $\langle\sigma, \tau\rangle$ is a semantic type. (Why not just say 'if **e** and **t** are semantic types, ...'?)

c. Nothing else is a semantic type.

Semantic denotation domains

a. $D_{\mathsf{e}} := D$ (**the set of** individuals)

b. $D_{\mathsf{t}} := \{0, 1\}$ (**the set of** truth-values)

c. For any semantic types σ and τ, $D_{\langle\sigma,\tau\rangle}$ is **the set of** all functions from D_σ to D_τ.

- So far, we've come across four denotation types:
  - type **e** (example: names)
  - type $\langle \mathbf{e}, \mathbf{t} \rangle$ (example: intransitive Vs)
  - type $\langle \mathbf{e}, \langle \mathbf{e}, \mathbf{t} \rangle \rangle$ (example: transitive Vs)
  - type **t** (example: sentences)

- We've covered a conceptually vast, yet relatively simple, metalinguistic system with(in) which we can analyse meanings.
- We now have **two more technical matters** to address:
  - One will decompose 2-place functions (=transitive Vs) and make sense of them in terms of the system we've been developing.
  - Another will simplify the technical issues with the way we've been writing down functions. It will make life easier. And it makes much sense.

SCHÖNFINKELISATION

- We need a bit more maths to synthesise the last portion of slides and understand trans-Vs as 2-place functions.
- Recall our three general assumptions:

Binary branching In the syntax, trans-Vs combine with the direct object to form a VP, and VPs combine with the subject to form a sentence.

Locality Semantic interpretation rules are local: the denotation of any non-terminal node is computed from the denotation of its daughter nodes.

Frege's conjecture Semantic composition is functional application.

Example

Let our domain D contain just the three goats Sebastian, Dimitri, and Leopold. Sebastian is the biggest and Leopold the smallest. The relation "is-bigger-than" is then the following set of ordered pairs:

$$(16) \quad R_{\text{BIGGER}} = \left\{ \begin{matrix} \langle \text{Sebastian, Dimitri} \rangle, \\ \langle \text{Sebastian, Leopold} \rangle, \\ \langle \text{Dimitri, Leopold} \rangle \end{matrix} \right\}$$

- There is a correspondence between sets and their characteristic functions.
- What is the *functional version* of R_{BIGGER}? (3mins)

- The resulting function f_{BIGGER} is a **2-place function**.
- Moses Schönfinkel, a logician, showed that n-place functions are **reducible to 1-place function**.
- This reduction is *Schönfinkelisation*.

$$f_{\text{BIGGER}} = \begin{bmatrix} \langle L,S\rangle \mapsto 0 \\ \langle L,D\rangle \mapsto 0 \\ \langle L,L\rangle \mapsto 0 \\ \langle S,L\rangle \mapsto 1 \\ \langle S,D\rangle \mapsto 1 \\ \langle S,S\rangle \mapsto 0 \\ \langle D,L\rangle \mapsto 1 \\ \langle D,S\rangle \mapsto 0 \\ \langle D,D\rangle \mapsto 0 \end{bmatrix} \longrightarrow f'_{\text{BIGGER}} = \begin{bmatrix} L \mapsto \begin{bmatrix} L \mapsto 0 \\ S \mapsto 0 \\ D \mapsto 0 \end{bmatrix} \\ S \mapsto \begin{bmatrix} L \mapsto 1 \\ S \mapsto 0 \\ D \mapsto 1 \end{bmatrix} \\ D \mapsto \begin{bmatrix} L \mapsto 1 \\ S \mapsto 0 \\ D \mapsto 0 \end{bmatrix} \end{bmatrix}$$

- f'_{BIGGER} is a function that applies to the first arg. and yields a function that applies to the second arg.
- When applied to Leopold, it yields a function that maps any goat to 1 **if it is smaller than Leopold**.

- We could also do it the other way round: have the function apply to the second argument and yield a function that applies to the first argument.
- Think of our syntactic tree.
- When applied to Leopold, let f'' yield a function that maps any goat to 1 **if it is bigger than Leopold**.

$$f'_{\text{BIGGER}} = \begin{bmatrix} \text{L} \mapsto \begin{bmatrix} \text{L} \mapsto 0 \\ \text{S} \mapsto 0 \\ \text{D} \mapsto 0 \end{bmatrix} \\ \text{S} \mapsto \begin{bmatrix} \text{L} \mapsto 1 \\ \text{S} \mapsto 0 \\ \text{D} \mapsto 1 \end{bmatrix} \\ \text{D} \mapsto \begin{bmatrix} \text{L} \mapsto 1 \\ \text{S} \mapsto 0 \\ \text{D} \mapsto 0 \end{bmatrix} \end{bmatrix} \longrightarrow f''_{\text{BIGGER}} = \begin{bmatrix} \text{L} \mapsto \begin{bmatrix} \text{L} \mapsto 0 \\ \text{S} \mapsto 1 \\ \text{D} \mapsto 1 \end{bmatrix} \\ \text{S} \mapsto \begin{bmatrix} \text{L} \mapsto 0 \\ \text{S} \mapsto 0 \\ \text{D} \mapsto 0 \end{bmatrix} \\ \text{D} \mapsto \begin{bmatrix} \text{L} \mapsto 0 \\ \text{S} \mapsto 1 \\ \text{D} \mapsto 0 \end{bmatrix} \end{bmatrix}$$

THE λ-CALCULUS

- We now turn to the second technical matter.
- We add some very special operators, **lambdas** (λ), to our system in order to simplify it.
- **The λ operator applies to a function in order to describe it.**

- Before we move onto this, let's recall our CHAR$_f$-notation for intransitive verbs.

(17) ⟦**Ann snores**⟧ = ⟦**snores**⟧(Ann) = 1 (iff Ann actually snores)

- What about transitive verbs?

(18) ⟦**Ann loves Jan**⟧ = (two notations)

a. ⟦**loves**⟧(Ann)(Jan) = 1 (iff Ann actually loves Jan)

b. ⟦**loves**⟧(Ann, Jan) = 1 (iff Ann actually loves Jan)

- Imagine we were interpreting an expression containing just the two words: noun **Maggie** and verb **love(s)**
  - We first need to construct a tree. In our case, there are two possible trees since something is missing.

(19)

λx.LOVES(Mary, x)

denotes the characteristic function of the set of individuals that **Maggie loves**.

(20)

λx.LOVES(x, Maggie)

denotes the characteristic function of the set of individuals that **love Maggie**.

- Very loosely, a λ-formula specifies the conditions that need to be met under which the function is true.
- A verb like **smoke** makes sense (it is or can be true) **only if there a single argument which can saturate its meaning**.

(21) $[\![\text{smokes}]\!] = \ldots$

(22) $\lambda x.[\![\text{smokes}]\!](x) = \ldots$

- The last notation can be read 'if there was an x, $[\![\text{smoke}]\!]$ could be true.'

- If φ is an expression denoting a function, and x is an expression that is of the right type to be used as an argument to φ, then $\varphi(x)$ denotes the result of applying φ to x (**saturation**).

For example

Expression BORED(x) **denotes the result of applying** the function denoted by **bored** to the value of x.

Another example

(23) $\left[\lambda x.\textsc{loves}(\text{Maggie})(x)\right](\text{Bill})$

- 23 denotes the result of applying the function **is loved by Maggie** to **Bill**.
- This is then equivalent to (24)

(24) $\textsc{loves}(\text{Maggie})(\text{Bill})$

- where **Bill** replaced the placeholder x.
- This 'conversion' process is known as **β-conversion** or **β-reduction**.

- We all remember formulae like (25) from high school.

(25) $f(x) = x + 7$

a. Now let $x = 5$.

b. Then we have:

$f(x) = x + 7 \rightsquigarrow f(5) = 5 + 7$

- (25) is the same as (27)

(26)

a. $f(x) = x + 7 \rightsquigarrow \lambda x.x + 7$

b. $[\lambda x.x + 7](5) \rightsquigarrow 5 + 7$

(27) a. $f(x) = x + 7 \rightsquigarrow \lambda x.x + 7$

b. $[\lambda x.x + 7](5) \rightsquigarrow 5 + 7$

- That's all λ-abstraction is:
  - abstraction with a λ-clause specifies the conditions under which the **value description** (27a)
  - β-reduction (β-conversion), **reduces** or **converts** the variable x into whatever value we feed it – in our case, number 5.

QUESTIONS?

EXERCISES

(28) $29 \in \{x \in \mathbb{N} : x \neq 0\}$ iff $29 \neq 0$

(29) Massachusetts $\in \{x \in D :$ California is a western state$\} = D$ iff California is a Western state.

(30) $\{x \in D :$ California is a western state$\} = D$ if California is a western state.

(31) $\{x \in D :$ California is a western state$\} = \varnothing$ if California is not a western state.

(32) $\{x \in \mathbb{N} : x \neq 0\} = \{y \in \mathbb{N} : y \neq 0\}$

exercise: simply the λ-expressions

(33) $[\lambda x \in D[\lambda y \in D[\lambda z \in D.z$ introduced x to $y]]](\text{Ann})(\text{Sue})$

(34) $[\lambda x \in \mathbb{N}[\lambda y \in \mathbb{N}.y > 3$ and $y < 7](x)]]]$