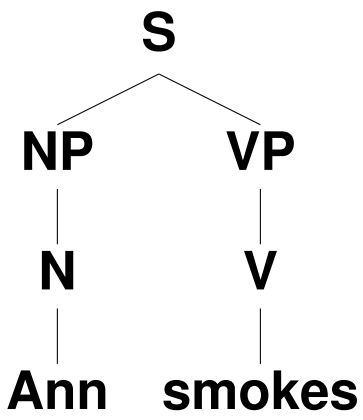
Executing the Fregean Program (Part I)

Heim and Kratzer

Chapter 2

First example of a Fregean Interpretation



- Frege took the denotation (extension) of a sentence to be a truth value: 1 (True) or 0 (False). Since the extensions of sentences are not functions, they are saturated.

- The extension of a proper name is just an individual (“Ann” denotes Ann), which is also saturated. The N and the NP nodes have the same extension.

- This means that the extension of a verb phrase must be unsaturated, a function from individuals to truth values. The V node and the intransitive verb have the same extension as the VP node.

First example of a Fregean Interpretation (cont.)

We now want to formalize our semantics for the fragment of English under consideration.

- We define our inventory of denotations, we provide a lexicon which specifies the denotation of each item that may occupy a terminal node, and we give a semantic rule for each possible type of non-terminal node.

- When we want to talk about the denotation of a lexical item or tree, we enclose it in double brackets: For any expression α, then $[\![\alpha]\!]$, is the denotation of α.

- We can think of $[\![\ ]\!]$ as a function (the *interpretation function*) that assigns appropriate denotations to linguistic expressions. For now, the denotations of expressions are extensions. In chapter 10, we will consider intensions.

First example of a Fregean Interpretation (cont.)

A. inventory of denotations

Let D be the set of all individuals that exist in the real world. Possible denotations are:

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

B. Lexicon

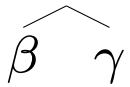
$[\![$**Ann**$]\!]$ = Ann
$[\![$**Jan**$]\!]$ = Jan
etc. for other proper names.

$[\![$**works**$]\!]$ = f : D $\rightarrow \{0, 1\}$
For all x $\in$ D, f(x) = 1 iff x works.
$[\![$**smokes**$]\!]$ = f : D $\rightarrow \{0, 1\}$
For all x $\in$ D, f(x) = 1 iff x smokes.
etc. for other intransitive verbs.

First example of a Fregean Interpretation (cont.)

C. Rules for non-terminal nodes

(S1) If α has the form



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

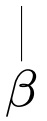
(S2) If α has the form

then $[\![\alpha]\!] = [\![\beta]\!]$.

(S3) If α has the form

then $[\![\alpha]\!] = [\![\beta]\!]$.

(S4) If α has the form



then $[\![\alpha]\!] = [\![\beta]\!]$.

(S5) If α has the form

then $[\![\alpha]\!] = [\![\beta]\!]$.

Applying the semantics to an example

- See text for proof that “Ann smokes” is true iff Ann smokes.
- Use each applicable rule or lexical entry to obtain an equation regarding the denotation of a certain subtree.
- Keep using some of these equations to substitute equals for equals in others, thereby getting closer and closer to the target equation in the claim.
- Employ the definitions of functions found in the lexicon to calculate their values for specified arguments.
- These steps can be performed in any order.
- We use “proof” semi-formally as it is used in mathematics: Proofs are written in plain English, supplemented by technical vocabulary that has been introduced through definitions.

Deriving truth conditions in an extensional semantics

- We end up with the truth-*conditions* of “Ann smokes” because the lexicon defines the extensions of predicates by specifying a *condition*.

- Had we defined the function denoted by “smokes” by displaying it in a table, we would have obtained a mere truth-*value*. But for that we would have to know some facts about the world, e.g. that:

$$[\![\text{smokes}]\!] = \begin{bmatrix} \text{Ann} \rightarrow 1 \\ \text{Jan} \rightarrow 1 \\ \text{Maria} \rightarrow 0 \end{bmatrix}$$

- Now we can calculate the truth value of our sentence:

$$\begin{bmatrix} \text{Ann} \rightarrow 1 \\ \text{Jan} \rightarrow 1 \\ \text{Maria} \rightarrow 0 \end{bmatrix} (\text{Ann}) = 1$$

Deriving truth conditions in an extensional semantics (cont.)

- The issue of how an extensional system can yield a theory of meaning concerns the relationship between what Frege called "*Sinn*" and "*Bedeutung*".

- Frege's "Bedeutung" corresponds to our term "extension" and is sometimes translated as "reference". Frege's "Sinn" is usually translated as "sense", and corresponds to what we have called "meaning".

- When specifying the extension (reference, *Bedeutung*) of an expression, we have to choose a particular way of presenting it, and it is this manner of presentation that might be considered its meaning (sense, *Sinn*).

- The function that is the extension of a predicate can be presented by providing a condition or by displaying it in a table, for example. Only if we provide a condition do we choose a mode of presentation that "shows" the meaning of the predicates and the sentences they occur in.

Object language and metalanguage

- When we have referred to words and phrases of English (represented as strings or trees), we replaced the customary quotes by bold-face. So we had “Ann” = **Ann**, for example.

- The expressions that are bold-faced or enclosed in quotes are expressions of our *object language*, the language we are investigating.

- The language we use for theoretical statements is the *metalanguage*. Given that this book is written in English, our metalanguage is English as well. The metalanguage includes a fair amount of technical vocabulary and notational conventions.

- Quotes or typographical distinctions help us mark the distinction between object language and metalanguage.

Object language and metalanguage (cont.)

- We always use bold-faced forms when we place object language expressions between brackets, e.g. [[**Ann**]] = Ann. This lexical entry determines that the denotation of the English name “Ann” is the person Ann.

- We will never write things like ”[[Ann]]”. This would have to be read as “the denotation of the person Ann”, which is nonsense. We also will not use bold-face for other purposes, such as emphasis, for which we use italics.

Sets and their characteristic functions

- We have construed the denotations of intransitive verbs as functions from individuals to truth values. Alternatively, they are often regarded as sets of individuals. The intuition is that each verb denotes the set of those things that it is true of. For example: $[\![\textbf{sleeps}]\!] = \{x \in D : x \text{ sleeps}\}$.

- This type of denotation would require a different semantic rule for composing subject and predicate that isn't simply function application.

- We have chosen to take Frege's Conjecture quite literally, but for some purposes, sets are easier to manipulate intuitively. It is therefore useful to be able to pretend in informal talk that intransitive verbs denote sets.

- Fortunately, this make-believe is harmless because there exists a one-to-one correspondence between sets and certain functions.

Sets and their characteristic functions (cont)

- Let A be a set. Then char_A, the *characteristic function* of A, is that function f such that, for any x $\in$ A, f(x) = 1, and for any x $\notin$ A, f(x) = 0.

- Let f be a function with range $\{0, 1\}$. Then char_f, the set characterized by f, is $\{x \in D : f(x) = 1\}$.

- We will often switch back and forth between function talk and set talk, sometimes saying things that are literally false, but become true when the references to sets are replaced by references to their characteristic functions (or vice versa).

Sets and their characteristic functions (cont)

- Using set talk, let's assume that $[\![\textbf{sleep}]\!] = \{\text{Ann, Jan}\}$, and $[\![\textbf{snore}]\!] = \{\text{Ann}\}$. We can now say things like:

- $\text{Ann} \in [\![\textbf{sleep}]\!]$.

- $[\![\textbf{snore}]\!] \subseteq [\![\textbf{sleep}]\!]$.

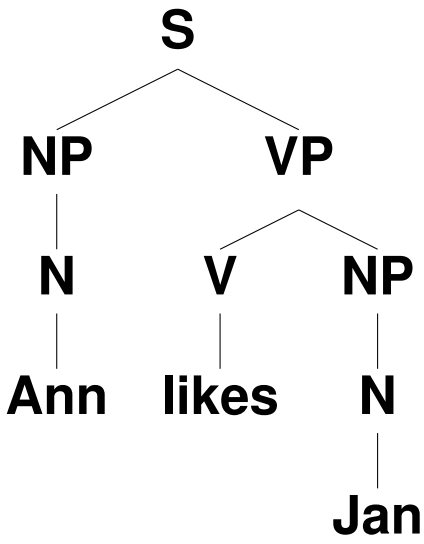
- $|[\![\textbf{snore}]\!] \cap [\![\textbf{sleep}]\!]| = 1$.

Sets and their characteristic functions (cont)

- Using function talk, we would have to say:
- $[\![\textbf{sleep}]\!](\text{Ann}) = 1$
- For all $x \in D$: if $[\![\textbf{snore}]\!](x) = 1$, then $[\![\textbf{sleep}]\!](x) = 1$
  Or equivalently,
  $\{x : [\![\textbf{snore}]\!](x) = 1\} \subseteq \{x : [\![\textbf{sleep}]\!](x) = 1\}$

- $|\{x : [\![\textbf{snore}]\!](x) = 1\} \cap \{x : [\![\textbf{sleep}]\!](x) = 1\}| = 1$

Adding transitive verbs: semantic types and denotation domains



- We know that NP nodes denote individuals and VP nodes denote functions from individuals to truth values. This means that transitive verb nodes denote *functions from individuals to functions from individuals to truth values*.

Adding transitive verbs: semantic types and denotation domains (cont)

[[**like**]] = f: D $\rightarrow$ {g : g is a function from D to {0, 1} }
For all x $\in$ D, f(x) = g_x : D $\rightarrow$ {0, 1}
For all y $\in$ D, g_x(y) = 1 iff y likes x.

Or, more succinctly:

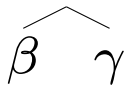
[[**like**]] = f: D $\rightarrow$ {g : g is a function from D to {0, 1} }
For all x, y $\in$ D, f(x)(y) = 1 iff y likes x.

Adding transitive verbs: semantic types and denotation domains (cont)

- [[**like**]] is a 1-place function, that is, a function with just one argument. The arguments of [[**like**]] are interpreted as the individuals which are liked; that is they correspond to the grammatical *object* of the verb “like”.

- This is so because we have assumed that transitive verbs form a VP with their direct object. The direct object, then, is the argument that is closest to a transitive verb, and is therefore semantically processed before the subject.

(S6) If α has the form VP then $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$.



Adding transitive verbs: semantic types and denotation domains (cont)

- It is convenient at this point to introduce a way of systematizing and labeling the types of denotations in our growing inventory. Following Montague, we employ “e” and “t” for the two basic types:

- e is the type of individuals.
  D_e := D

- t is the type of truth values.
  D_t := $\{0, 1\}$

- $D_{<e,t>}$:= $\{$f : f is a function from D_e to $D_t\}$

- $D_{<e,<e,t>>}$:= $\{$f : f is a function from D_e to $D_{<e,t>}\}$

Adding transitive verbs: semantic types and denotation domains (cont)

Semantic types

(a) e and t are semantic types.

(b) if σ and τ are semantic types, then $< \sigma, \tau >$ is a semantic type.

(c) Nothing else is a semantic type.

Semantic denotation domains

(a) D_e := D (the set of individuals)

(b) D_t := $\{0, 1\}$ (the set of truth values)

(c) For any semantic types σ and τ , $D_{<\sigma, \tau>}$ is the set of all functions from D_σ to D_τ

Adding transitive verbs: semantic types and denotation domains (cont)

- Which semantic types are actually used by natural languages is still a matter of debate. The issue of "type economy" will arise throughout the book, most notably for adjectives and quantifier phrases.

- We have encountered terminal nodes of type e (proper names), type $< e, t >$ (intransitive verbs) and type $< e, < e, t >>$ (transitive verbs).

- We have encountered non-terminal nodes of type e (N, NP), type $< e, t >$ (VP and certain Vs), type $< e, < e, t >>$ (the other Vs), and type t (S).

Schönfinkelization

Denotations of transitive verbs must be 1-place functions. This follows from three assumptions about the syntax-semantics interface:

- **Binary Branching**: In the syntax, transitive verbs 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 denotations of its daughter nodes.

- **Frege's Conjecture**: Semantic composition is functional application.

Schönfinkelization (cont.)

In logic texts, the extension of a 2-place predicate is a set of ordered pairs.

- For example, let D = {Fiona, Patsy, Jenny}
  [[**like**]]= $\{<$ Fiona, Patsy $>, <$ Patsy, Jenny $>, <$ Jenny, Jenny $>\}$

- The characteristic function of [[**like**]] =

$$\begin{bmatrix} <\text{Fiona, Fiona}> \rightarrow 0 \\ <\text{Fiona, Patsy}> \rightarrow 1 \\ <\text{Fiona, Jenny}> \rightarrow 0 \\ <\text{Patsy, Fiona}> \rightarrow 0 \\ <\text{Patsy, Patsy}> \rightarrow 0 \\ <\text{Patsy, Jenny}> \rightarrow 1 \\ <\text{Jenny, Fiona}> \rightarrow 0 \\ <\text{Jenny, Patsy}> \rightarrow 0 \\ <\text{Jenny, Jenny}> \rightarrow 1 \end{bmatrix}$$

Schönfinkelization (cont.)

Schönfinkel showed how n-place functions can be reduced to 1-place functions:

Left-to-right

$$\left[\begin{array}{l} \text{Fiona} \rightarrow \left[\begin{array}{l} \text{Fiona} \rightarrow 0 \\ \text{Patsy} \rightarrow 1 \\ \text{Jenny} \rightarrow 0 \end{array}\right] \\ \text{Patsy} \rightarrow \left[\begin{array}{l} \text{Fiona} \rightarrow 0 \\ \text{Patsy} \rightarrow 0 \\ \text{Jenny} \rightarrow 1 \end{array}\right] \\ \text{Jenny} \rightarrow \left[\begin{array}{l} \text{Fiona} \rightarrow 0 \\ \text{Patsy} \rightarrow 0 \\ \text{Jenny} \rightarrow 1 \end{array}\right] \end{array}\right]$$

Right-to-left

$$\left[\begin{array}{l} \text{Fiona} \rightarrow \left[\begin{array}{l} \text{Fiona} \rightarrow 0 \\ \text{Patsy} \rightarrow 0 \\ \text{Jenny} \rightarrow 0 \end{array}\right] \\ \text{Patsy} \rightarrow \left[\begin{array}{l} \text{Fiona} \rightarrow 1 \\ \text{Patsy} \rightarrow 0 \\ \text{Jenny} \rightarrow 0 \end{array}\right] \\ \text{Jenny} \rightarrow \left[\begin{array}{l} \text{Fiona} \rightarrow 0 \\ \text{Patsy} \rightarrow 1 \\ \text{Jenny} \rightarrow 1 \end{array}\right] \end{array}\right]$$

Which Schönfinkelization is consistent with the principle of compositional semantics? Left-to-right or Right-to-left?

Schönfinkelization (cont.)

- it is the right-to-left Schönfinkelization that corresponds to the Fregean denotation of a 2-place predicate because the corresponding relations are customarily specified so that the grammatical object corresponds to the right component of each pair in the relation, and the subject to the left one. This is an arbitrary convention in a way.

- It is not arbitrary that the unique argument of the Fregean denotation of a 2-place predicate corresponds to the grammatical object. Since the object is closest to the predicate in hierarchical terms, it must provide the argument for the function denoted by the predicate.