Solutions to Exercises for Heim and Kratzer (1997): *Semantics in Generative Grammar*

Miguel Hoeltje

1. Truth-conditional Semantics and the Fregean Program

Exercise 1.1 (Chapter 1, p.9)
The same set can be described in many different ways, often quite different superficially. Here you are supposed to figure out which of the following equalities hold and which ones don't. Sometimes the right answer is not just plain "yes" or "no", but something like "yes, but only if ...". For example, the two sets in (i) are equal only in the special case where $a = b$. In case of doubt, the best way to check whether two sets are equal is to consider an arbitrary individual, say John, and to ask if John could be in one of the sets without being in the other as well.

a) $\{a\} = \{b\}$
b) $\{x : x = a\} = \{a\}$
c) $\{x : x \text{ is green}\} = \{y : y \text{ is green}\}$
d) $\{x : x \text{ likes } a\} = \{y : y \text{ likes } b\}$
e) $\{x : x \in A\} = A$
f) $\{x : x \in \{y : y \in B\}\} = B$
g) $\{x : \{y : y \text{ likes } x\} = \varnothing\} = \{x : \{x : x \text{ likes } x\} = \varnothing\}$

Solution 1.1

a) True if and only if $a = b$.
b) True.
c) True.
d) True if and only if a and b are liked by the same people.
e) True.
f) True.
g) This one is tricky! Don't worry if you didn't solve this one.

Let us use A for the first set $\{x : \{y : y \text{ likes } x\} = \varnothing\}$, and B for the second set $\{x : \{x : x \text{ likes } x\} = \varnothing\}$. A is the set of all people who no one likes (more long winded: it is the set of all x who are such that the set of all y who like x is empty).

The set B is more confusing. There are (at least) two general things one has to note to understand what B actually is. *First*, if a formula Φ does not contain any free variables, then '$\{x : \Phi\}$' will either denote the universal set (if Φ is true), or the empty set (if Φ is false). (Heim & Kratzer illustrates the first case with the example '$\{x : \text{California is a western state}\}$' in section 1.3.2.) *Second*, there are no free variables in '$\{x : x \text{ likes } x\} = \varnothing$' (since the '$x$' before the ':' binds those in 'x likes x'). Thus, B is either the universal set or the empty set, depending on whether '$\{x : x \text{ likes } x\} = \varnothing$' is true or false.

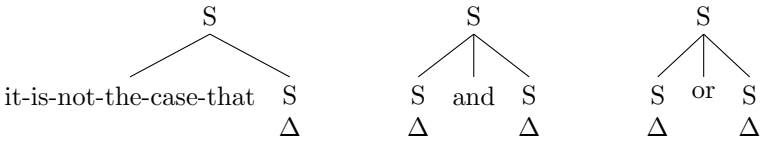
But which truth-value does '$\{x : x \text{ likes } x\} = \varnothing$' have? The formula says that the set of all x such that x likes x is empty; or, in other words, that nobody likes themselves. Thus, if nobody likes themselves, B is the universal set; and if at least somebody likes themselves, B is the empty set. (Another way of describing B is as the set of all people which are such that no one likes themselves; compare the California-example above.)

Now the question: under which circumstances are A (the set of all people who no one likes) and B (the set of all people which are such that no one likes themselves) identical? This is the case iff either of the following two conditions hold: *(i)* Nobody likes anybody (in this case 'both' A and B contain everything); *(ii)* Someone likes themselves *and* everyone is liked by someone (in this case 'both' A and B are empty).

Phew. I think we got it. Any errors?

2. Executing the Fregean Program

Exercise 2.1 (Chapter 2, p.23)
Suppose we extend our fragment to include phrase structures of the forms below (where the embedded S-constituents may either belong to the initial fragment or have one of these three forms themselves):



How do we have to revise and extend the semantic component in order to provide all the phrase structures in this expanded fragment with interpretations? Your task in this exercise is to define an appropriate semantic value for each new lexical item (treat "it-is-not-the-case-that" as a single lexical item here [below abbreviated as *not*) and to write appropriate semantic rules for the new types of non-terminal nodes. To do this, you will also have to expand the inventory of possible semantic values. Make sure that you stick to our working hypothesis that all semantic composition is functional application (Frege's Conjecture).

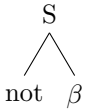
Solution 2.1
Recall that in chpt. 2, Heim and Kratzer construe a semantic theory as consisting of three components: first, an *inventory of possible denotations*; second, a *lexicon* which specifies the denotations of lexical items (i.e. items that may occupy terminal nodes); third, a collection of *semantic composition rules* for possible types of non-terminal nodes. Accordingly, our solution will proceed in three steps.

First, in order to assign appropriate semantic values to sentential operators such as *not*, *and*, and *or*, we need to expand the inventory of possible denotations. In particular, we will allow *functions from truth values to truth values* (in order to treat one-placed operators such as *not*), and we will allow functions from *pairs of truth values* to truth values (in order to treat two-placed operators such as *and* and *or*). We will use D_t to refer to the set of truth values: $D_t = \{0, 1\}$. Accordingly, we will use $D_t \times D_t$ to refer to the set of *pairs* of truth values. Our new inventory of denotions thus encompasses functions from D_t to D_t, as well as functions from $D_t \times D_t$ to D_t.

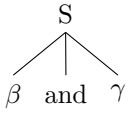
In the *second* step, we add three new entries to the lexicon which specify the semantic values for *not*, *and*, *or*:

$[\![\text{not}]\!] = f : D_t \to D_t$
For all $x \in D_t, f(x) = 1 \leftrightarrow x = 0$.
$[\![\text{and}]\!] = f : D_t \times D_t \to D_t$
For all $\langle x, y\rangle \in (D_t \times D_t), f(\langle x, y\rangle) = 1 \leftrightarrow (x = 1 \wedge y = 1)$.
$[\![\text{or}]\!] = f : D_t \times D_t \to D_t$
For all $\langle x, y\rangle \in (D_t \times D_t), f(\langle x, y\rangle) = 1 \leftrightarrow (x = 1 \vee y = 1)$.

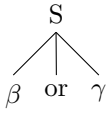
In the *third* step, we need to provide composition rules which cover all phrase structures of the forms presented in the exercise. We will use α, β, and γ as variables for trees and subtrees. The first new rule covers phrase structures formed by negation:

If α has the form  then $[\![\alpha]\!] = [\![\text{not}]\!]([\![\beta]\!])$.

The second rule covers phrase structures involving *and* as a main operator:

If α has the form  then $[\![\alpha]\!] = [\![\text{and}]\!](\langle[\![\beta]\!],[\![\gamma]\!]\rangle)$.

The third rule covers phrase structures involving *or* as a main operator (note that this rule structurally exactly mirrors the rule for *and*):

If α has the form  then $[\![\alpha]\!] = [\![\text{or}]\!](\langle[\![\beta]\!],[\![\gamma]\!]\rangle)$.

Exercise 2.2 (Section 2.2, p.24)

We have construed the denotations of intransitive verbs as functions from individuals to truth-values. Alternatively, they are often regarded as sets of individuals. This is the standard choice for the extensions of 1-place predicates in logic. The intuition here is that each verb denotes the set of those things that it is true of. For example: $[\![\text{sleeps}]\!] = \{x \in D : x \text{ sleeps}\}$. This type of denotation would require a different semantic rule for composing subject and predicate, one that isn't simply functional application. Write the rule it would require.

Solution 2.2

If α has the form  then $[\![\alpha]\!] = 1 \leftrightarrow [\![\beta]\!] \in [\![\gamma]\!]$.

Exercise 2.3 (Section 2.4, p.31)

Suppose that our universe D contains just two elements, Jacob and Maria. Consider now the following binary and ternary relations:

R_{adores} = {⟨Jacob, Maria⟩, ⟨Maria, Maria⟩}
$R_{\text{assigns to}}$ = {⟨Jacob, Jacob, Maria⟩, ⟨Maria, Jacob, Maria⟩}

In standard predicate logic, these would be suitable extensions for the 2-place and 3-place predicate letters F^2 and G^3 as used in the following scheme of abbreviation:

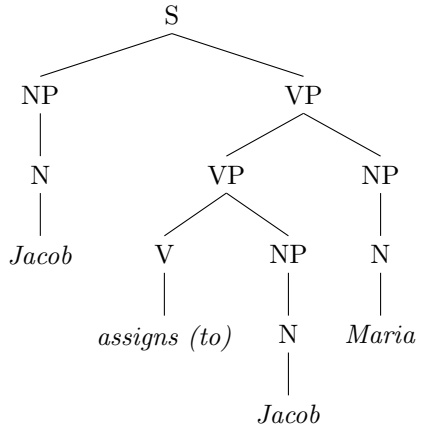
F^2: a adores b.
G^3: a assigns b to c.

(i) Find the characteristic functions for both of these relations, and then Schönfinkel them from right to left. *(ii)* Could the two Schönfinkeled functions be suitable denotations for the English verbs *adore* and *assign (to)* respectively? If yes, why? If not, why not?

Clarifying remarks on part (i) of the exercise (added by MH): Heim and Kratzer introduce Schönfinkelization in §2.4 by illustrating it only with respect to 2-place functions. For the purposes of this exercise, assume that the right-to-left Schönfinkelization f'' of a 3-place function f (i.e. of a function f whose domain consists of ordered triples) is such that the following holds: f'' maps x to a function which maps y to a function which maps z to α if and only if f maps the triple $\langle z, y, x\rangle$ to α.

Clarifying remarks on part (ii) of the exercise (added by MH): In order to solve part *(ii)* of this exercise, you need to assume some branching structure for sentences such as *Jacob adores Maria* and *Jacob assigns Jacob to Maria*. For the first, assume that it has a branching structure corresponding to the one that Heim and Kratzer give for *Ann likes Jan* on p.26. For the second, treat *assigns (to)* as syntactically primitive (see exercise 2.5 for a more elaborate syntactic structure), and assume that the verb *assigns (to)* combines first with its direct object (*Jacob*), second with its indirect object (*Maria*), and finally with its subject (*Jacob* again). More specifically, assume the following branching structure for *Jacob assigns Jacob to Maria*:[1]

[1] Note that Heim and Kratzer's way of drawing syntactic trees differs in at least one respect from how more recent works in syntax often proceed. In the following tree for *Jacob adores Maria*, Heim and Kratzer take a lexical item such as *Jacob* occupy its own (terminal) node which is dominated by a non-terminal N node. This contrasts with how standard introductions to syntax construe the relation between terminal nodes and lexical items. For instance, Carnie 2013 would give a tree for *Jacob adores Maria* which terminates in the nodes N, V, and N (where the lexical items are written directly below these categories with no line between them). As Carnie writes: 'There are no phrase structure rules that connect words with their categories (i.e., there is no rule [V → *adores*]), so technically speaking any line between the word's category and the word is incorrect'. In the interest of uniformity, we will follow Heim and Kratzer's notation.



Solution 2.3

(i) Let us write J and M for *Jacob* and *Maria*, respectively. The characteristic function f_{adores} for R_{adores} is a function from pairs of objects to truth values. More specifically, it is the function which maps the pairs $\langle J, M\rangle$ and $\langle M, M\rangle$ to 1, and all other pairs to 0:

$f_{\text{adores}} = f : D_e \times D_e \to D_t$
For all $x \in (D_e \times D_e), f(x) = 1 \leftrightarrow (x = \langle J, M\rangle \vee x = \langle M, M\rangle)$.

We can also specify f_{adores} in a table:

$$f_{\text{adores}} = \begin{bmatrix} \langle J, J\rangle & \to & 0 \\ \langle J, M\rangle & \to & 1 \\ \langle M, J\rangle & \to & 0 \\ \langle M, M\rangle & \to & 1 \end{bmatrix}$$

There are two ways to Schönfinkel a two-placed function such as f_{adores}: from left to right, and from right to left (see p.30).[2] The right-to-left Schönfinkelization is f''_{adores}:

$$f''_{\text{adores}} = \begin{bmatrix} J & \to & \begin{bmatrix} J & \to & 0 \\ M & \to & 0 \end{bmatrix} \\ M & \to & \begin{bmatrix} J & \to & 1 \\ M & \to & 1 \end{bmatrix} \end{bmatrix}$$

The characteristic function $f_{\text{assigns to}}$ for $R_{\text{assigns to}}$ is a function from triples of objects to truth values. More specifically, it is the function which maps the triples $\langle J, J, M\rangle$ and $\langle M, J, M\rangle$ to 1, and everything else to 0:

$f_{\text{assigns to}} = f : D_e \times D_e \times D_e \to D_t$
For all $x \in (D_e \times D_e \times D_e), f(x) = 1 \leftrightarrow (x = \langle J, J, M\rangle \vee$
$x = \langle M, J, M\rangle)$.

We can also specify $f_{\text{assigns to}}$ in a table:

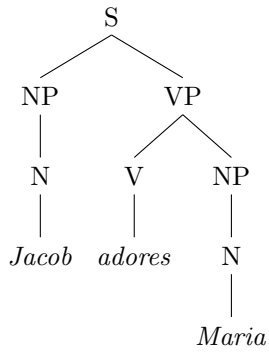
$$f_{\text{assigns to}} = \begin{bmatrix} \langle J, J, J\rangle & \to & 0 \\ \langle J, J, M\rangle & \to & 1 \\ \langle J, M, J\rangle & \to & 0 \\ \langle J, M, M\rangle & \to & 0 \\ \langle M, J, J\rangle & \to & 0 \\ \langle M, J, M\rangle & \to & 1 \\ \langle M, M, J\rangle & \to & 0 \\ \langle M, M, M\rangle & \to & 0 \end{bmatrix}$$

[2] We can define the left-to-right Schönfinkelization of a 2-place function f as $f' = \{\langle x, C\rangle : C = \{\langle y, z\rangle : f(\langle x, y\rangle) = z\}\}$. Correspondingly, the right-to-left Schönfinkelization of f is $f'' = \{\langle x, C\rangle : C = \{\langle y, z\rangle : f(\langle y, x\rangle) = z\}\}$.

While f_{adores} is a two-placed function, $f_{\text{assigns to}}$ is a *three*-placed function (i.e. a function which takes triples as input). The right-to-left Schönfinkelization f'' of a three-placed function f will behave as follows: If f maps $\langle x, y, z\rangle$ to a, then f'' will map z to a function which maps y to a function which maps x to a. Hence, the right-to-left Schönfinkelization of $f_{\text{assigns to}}$ is:

$$f''_{\text{assigns to}} = \begin{bmatrix} J \rightarrow \begin{bmatrix} J \rightarrow \begin{bmatrix} J \rightarrow 0 \\ M \rightarrow 0 \end{bmatrix} \\ M \rightarrow \begin{bmatrix} J \rightarrow 0 \\ M \rightarrow 0 \end{bmatrix} \end{bmatrix} \\ M \rightarrow \begin{bmatrix} J \rightarrow \begin{bmatrix} J \rightarrow 1 \\ M \rightarrow 1 \end{bmatrix} \\ M \rightarrow \begin{bmatrix} J \rightarrow 0 \\ M \rightarrow 0 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

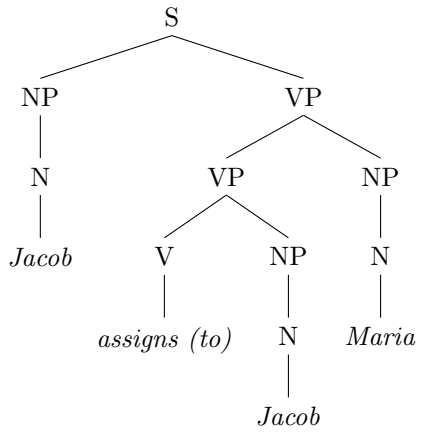
(ii) Let us start with the verb *adores*. Yes, the right to left Schönfinkelization f''_{adores} can be used as denotation for the verb *adores*. On p.26, Heim and Kratzer assumed that a sentence such as *Jacob adores Maria* has the following syntactic structure:



According to rule (S1) for S nodes given on p.16, the denotation of the S *Jacob adores Maria* is computed by applying the denotation of the VP *adores Maria* to the denotation of the NP *Jacob*: ⟦Jacob adores Maria⟧ = ⟦adores Maria⟧(⟦Jacob⟧). Moreover, according to rule (S6) for branching VP nodes given on p.27, the denotation of the VP *adores Maria* is computed by applying the denotation of the V *adores* to the denotation of the NP *Maria*: ⟦adores Maria⟧ = ⟦adores⟧(⟦Maria⟧). Putting these two things together, we derive: ⟦Jacob adores Maria⟧ = (⟦adores⟧(⟦Maria⟧)) (⟦Jacob⟧). We can now see that assuming ⟦adores⟧ = f''_{adores} yields the correct denotations for the relevant sentences (check the flow chart for f''_{adores} above). More generally, the right-to-left Schönfinkelization works in the case of a transitive verb such as *adores* since we have assumed that 'transitive verbs combine with the direct object to form a VP, and VPs combine with the subject to form a sentence' (see the assumption entitled *Binary Branching* on p.29).[3]

Let us now turn to the verb *assigns (to)*. Here the answer is: No, the right-to-left Schönfinkelization $f''_{\text{assigns to}}$ cannot be used as denotation for the verb *assigns (to)* (under our assumptions about the branching structure of the relevant sentences). To repeat, here is the structure we assumed for *Jacob assigns Jacob to Maria*:

[3]The title *Binary Branching* for this assumption is perhaps a little misleading; strictly speaking, this assumption is the conjunction of the claim that we have binary branching and the claim that transitive verbs combine with their direct object first. In principle, the mere claim that we have binary branching is also be compatible with the additional claim that transitive verbs combine with their subjects first (i.e. that we have [[Jacob adores] [Maria]] rather than [[Jacob][adores Maria]]). Note that if this was what our syntax told us, we would have to use the left-to-right Schönfinkelization of f_{adores} in our semantics.



According to the rules for non-terminal nodes (rules (S1)-(S5) on p.16 and rule (S6) on p.27), we have:

$[\![J \text{ assigns } J \text{ to } M]\!] = \Big(\big([\![\text{assigns (to)}]\!]\ ([\![J]\!])\big)\ ([\![M]\!])\Big)\ ([\![J]\!])$

In other words, we need to compute [[Jacob assigns Jacob to Maria]] by first applying [[assigns (to)]] to Jacob; second applying the resulting function to Maria; and finally applying the resulting function to Jacob again. We should end up with the truth value 1; after all, it was part of the setup that $\langle J, J, M\rangle \in R_{\text{assigns to}}$. However, the right-to-left Schönfinkelization of $f_{\text{assigns (to)}}$ produces the opposite result. As can be seen from inspecting the flowchart for f'' above, applying f'' first to J(acob), applying the result to M(aria), and applying the result again to J(acob), we end up with a 0, not with the desired 1.

The underlying problem can be stated as follows. In standard predicate logic, we construe the extension of the 3-place predicate *x assigns y to z* as a set of triples, where this set contains a triple $\langle x, y, z\rangle$ if and only if x assigns y to z. Accordingly, the right-to-left Schönfinkelization of the characteristic function of such a set will map z to a function which maps y to a function which maps x to the truth value 1 if and only if x assigns y to z. However, on our assumptions about the branching structure of sentences of the form 'x assigns y to z', we need a function which maps y to a function which maps z to a function which maps x to the truth value 1 if and only if x assigns y to z. Hence, the right-to-left Schönfinkelization of $f_{\text{assigns to}}$ gets the order of the arguments wrong.[4]

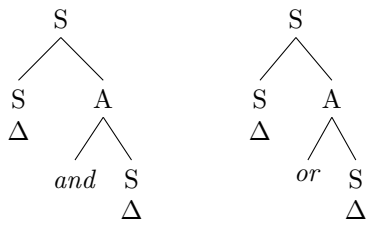
Exercise 2.4 (Chapter 2, p.32)

(i) In the exercise on sentence connectives in section 2.1, we stipulated ternary branching structures for sentences with *and* and *or*. Now assume that all English phrase structures are at most binary branching, and assign accordingly revised syntactic analyses to these sentences. (Whether you choose right-branching or left-branching structures does not matter here, but stick to one option.) Then revise the semantics accordingly. As always, be sure to provide every subtree with a semantic value, as well as to adhere to our current assumptions about the semantic interpretation component (Locality and Frege's Conjecture).

(ii) Using the labelling system introduced at the end of section 2.3, specify the type of denotation for each node in your binary branching structure for the sentence *Jan works, and it is not the case that Jan smokes.*

Solution 2.4

(i) We will solve this exercise in three steps. *First*, we present the new syntactic analyses. We chose right-branching and assign the following syntactic analyses:[5]



[4] Note that the left-to-right Schönfinkelization $f'_{\text{assigns to}}$ can *also* not be used as a denotation of *assigns (to)*. The left-to-right Schönfinkelization f' of a three-placed function f will behave as follows: If f maps $\langle x, y, z\rangle$ to a, then f' will map x to a function which maps y to a function which maps z to a. We will return to this issue exercise 2.5.

[5] As far as I can see, Heim and Kratzer do not suggest syntactic categories for *not*, *and*, and *or* (nor for the phrases formed by applying *and* and *or* to a first sentence). Since syntax is not our focus, we will not go into this issue and simply use the label *A* for whatever syntactic category should be assigned to phrases of the form *and/or S*.

In the *second* step, we revise the denotations of *and* and *or* accordingly:

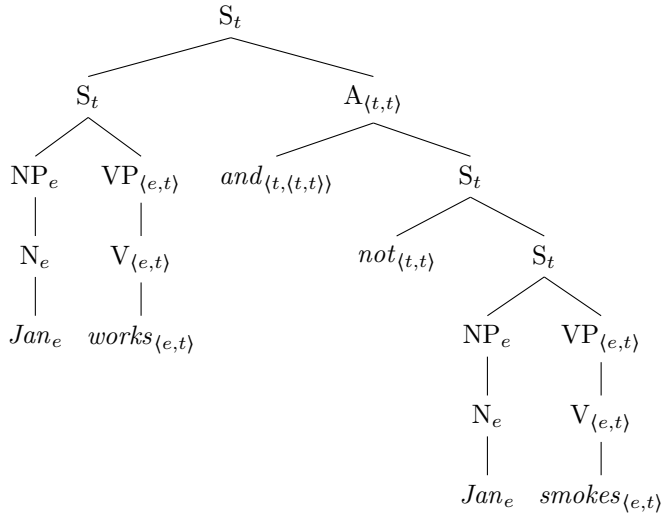
$[\![\text{and}]\!] = f : D_t \to \{g : g \text{ is a function from } D_t \text{ to } D_t\}$
For all $x, y \in D_t$, $f(x)(y) = 1$ iff ($y = 1$ and $x = 1$).
$[\![\text{or}]\!] = f : D_t \to \{g : g \text{ is a function from } D_t \text{ to } D_t\}$
For all $x, y \in D_t$, $f(x)(y) = 1$ iff ($y = 1$ or $x = 1$).

In the *third* step, we provide a new rule for the new kind of non-terminal node:

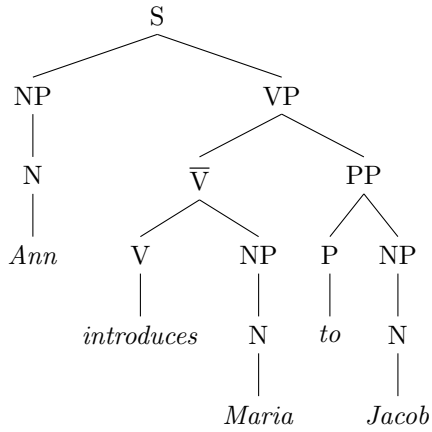
If α has the form  then $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$.

Note that this new rule does roughly the same as the rule (S1) (see p.16), only it reverses which daughter is taken to provide the function and which is taken to provide the argument. Had we chosen a left branching structure above, we would not have needed to give a new rule in addition to (S1).[6]

(ii) We will assume the branching structure indicated below and specify the types of denotation by subscripts:

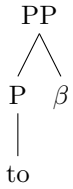


Exercise 2.5 (Chapter 2, p.32)
(a) Extend the fragment in such a way that phrase structure trees of the following kind are included:



Add the necessary semantic rules and lexical entries, sticking to Locality and Frege's Conjecture. Assume that the preposition *to* is a semantically vacuous element; that is, assume the ad hoc rule below:

[6] To foreshadow future discussion somewhat: Note also that the principle of *Functional Application* to be introduced on p.44 in a sense combines the above rule and (S1), and generalizes the result to trees of arbitrary syntactic categories.

If α has the form  then $[\![\alpha]\!] = [\![\beta]\!]$.

(b) Suppose now that the actual world contains just three individuals, Ann, Maria, and Jacob. And suppose further that Ann introduces Maria to Jacob, and Maria introduces Jacob to Ann, and no further introductions take place. Which particular function is [[introduce]] in this case? Display it in a *table*.
(c) Using the table specification of [[introduce]] from *(b)* and the lexical entries for the names, calculate the denotations of each non-terminal node in the tree under *(a)*.
(d) Under standard assumptions, a predicate logic formalization of the English sentence *Ann introduces Maria to Jacob* might look like this:

$$I^3(A\,M\,J)$$

Scheme of abbreviation:

A: *Ann* M: *Maria* J: *Jacob* I^3: *a introduces b to c*

The extension of I^3 under this scheme of abbreviation is the following set X of ordered triples:

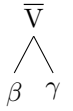
$X = \{\langle x, y, z\rangle \in D \times D \times D : x \text{ introduces } y \text{ to } z\}$

How is this extension related to the extension—let's call it f—for *introduce* that you defined in *(a)*? Give your answer by completing the following statement: For any x, y, $z \in D$, $f(x)(y)(z) = 1$ iff $\ldots \in X$.

Solution 2.5

(a) First we need to give a new semantic rule which takes care of the $\overline{\text{V}}$ node in the phrase structure tree for *Ann introduces Maria to Jacob*. Since we are assuming Locality and Frege's Conjecture, we know that these rules must determine the denotation of the respective node by applying the denotation of one of the daughters of the node to the denotation of the other daughter via functional application. Hence, in order to specify the rules, let us first figure out the semantic types of the various nodes in our tree.

Let us go through the tree starting from the root S. Rule (S1) from p.16 tells us that the denotation of the S node is computed by applying the denotation of the VP node to the denotation of the NP node. Rules (S2) and (S4) together tell us that the NP node gets its denotation from the lexical item *Ann*; hence we know that the denotation of the NP node is of type e. Since the S node has a denotation of type t, the VP node must therefore be of type $\langle e, t\rangle$. From the ad hoc rule stipulated in the exercise, we know that the denotation of the PP node is simply the denotation of the NP node below, which in turn is the denotation of the lexical item *Jacob* (rules (S2) and (S4) again). Hence, the PP node has a denotation of type e. Given Locality and Frege's Conjecture, we can thus infer that the $\overline{\text{V}}$ node must be of type $\langle e, \langle e, t\rangle\rangle$. But since we know that the NP node below it is of type e (rules (S2) and (S4) again), we can infer that the V node must be of type $\langle e, \langle e, \langle e, t\rangle\rangle\rangle$. Finally, by rule (S5), we infer that the lexical item *introduces* must have a denotation of type $\langle e, \langle e, \langle e, t\rangle\rangle\rangle$. We can now see that the rule for the $\overline{\text{V}}$ node must be the following rule, which is structurally identical to rule (S6) given for VPs on p.27:

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

Let us now turn to the lexical entries. Since we have assumed an ad hoc rule for *to*, and since all other expressions have been covered previously, we can make do with a lexical entry for *introduces*:

$[\![\text{introduces}]\!] = f : D_e \to \{g : g \text{ is a function from } D_e \text{ to functions from } D_e \text{ to } D_t\}$
For all $x, y, z \in D_e$, $f(x)(y)(z) = 1$ iff z introduces x to y.

(b) The relevant table is:

$$[\![\text{introduce}]\!] = \begin{bmatrix} A \to \begin{bmatrix} A \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ M \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ J \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \end{bmatrix} \\ M \to \begin{bmatrix} A \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ M \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ J \to \begin{bmatrix} A \to 1 \\ M \to 0 \\ J \to 0 \end{bmatrix} \end{bmatrix} \\ J \to \begin{bmatrix} A \to \begin{bmatrix} A \to 0 \\ M \to 1 \\ J \to 0 \end{bmatrix} \\ M \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ J \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(c) The denotations of the relevant non-terminal nodes are as follows:

$$\overline{\text{V}} : [\![\text{introduces Maria}]\!] = \begin{bmatrix} A \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ M \to \begin{bmatrix} A \to 0 \\ M \to 0 \\ J \to 0 \end{bmatrix} \\ J \to \begin{bmatrix} A \to 1 \\ M \to 0 \\ J \to 0 \end{bmatrix} \end{bmatrix}$$

$$\text{VP} : [\![\text{introduces Maria to Jacob}]\!] = \begin{bmatrix} A \to 1 \\ M \to 0 \\ J \to 0 \end{bmatrix}$$

$$\text{S} : [\![\text{Ann introduces Maria to Jacob}]\!] = 1$$

(d) For any x, y, $z \in D$, $f(x)(y)(z) = 1$ iff $\langle z, x, y \rangle \in X$.

Exercise 2.6 (Chapter 2, p.39)
Describe the following functions in words:

a) $\lambda x \in \mathbb{N}\,.\ x > 3$ and $x < 7$

b) $\lambda x : x$ is a person$\,.\ x$'s father

c) $\lambda X \in \mathcal{P}(D)\,.\ \{y \in D : y \notin X\}$

d) $\lambda X \subseteq D\,.\ [\lambda y \in D\,.\ y \notin X]$

Solution 2.6

a) The smallest function which maps any natural number greater than 3 and smaller than 7 to the truth value 1, and every other natural number to the truth value 0. (The characteristic function of $\{4, 5, 6\}$.)

b) The smallest function which maps any person x to the father of x.

c) $\mathcal{P}(D)$ is the set of all subsets of D, which we call *the powerset of* D. X's complement in D is the set of things that are in D but *not* in X. Hence, we can say that the function is the following: The smallest function which maps any subset X of D to X's complement in D.

d) The smallest function which maps any subset X of D to the characteristic function of X's complement in D.

Exercise 2.7 (Chapter 2, p.39)

In this exercise, simple functions are described in a rather complicated way. Simplify the descriptions as much as possible:

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

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

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

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

e) $\big[\lambda f \in D_{\langle e,t\rangle}\,.\ [\lambda x \in D_e\,.\ f(x) = 1 \text{ and } x \text{ is grey}]\big]\,([\lambda y \in D_e\,.\ y \text{ is a cat}])$

f) $\big[\lambda f \in D_{\langle e,\langle e,t\rangle\rangle}\,.\ [\lambda x \in D_e\,.\ f(x)(\text{Ann}) = 1]\big]\,([\lambda y \in D_e\,.\ [\lambda z \in D_e\,.\ z \text{ saw } y]])$

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

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

Solution 2.7

The simplified function descriptions are:

a) $[\lambda z \in D\,.\ z \text{ introduced Ann to Sue}]$

b) $[\lambda x \in D\,.\ \text{Ann introduced } x \text{ to Sue}]$

c) $[\lambda y \in D\,.\ \text{Ann introduced Sue to } y]$

d) $[\lambda z \in D\,.\ z \text{ introduced Sue to Ann}]$

e) $[\lambda x \in D_e\,.\ x \text{ is a cat and } x \text{ is grey}]$

f) $[\lambda x \in D_e\,.\ \text{Ann saw } x]$

g) $[\lambda x \in \mathbb{N}\,.\ x > 3 \text{ and } x < 7]$

h) $[\lambda z \in \mathbb{N}\,.\ z > 3 \text{ and } z < 7]$

Exercise 2.8 (Chapter 2, p.40)

Suppose *and* and *or* have Schönfinkeled denotations, that is $[\![\text{and}]\!]$ and $[\![\text{or}]\!]$ are both members of $D_{\langle t,\langle t,t\rangle\rangle}$. They are functions that map truth-values into functions from truth-values to truth-values. Specify the two functions using the λ-notation.

Solution 2.8

$[\![\text{and}]\!] = \lambda p \in D_t\,.\ [\lambda q \in D_t\,.\ p = 1 \text{ and } q = 1]$
$[\![\text{or}]\!] \ = \lambda p \in D_t\,.\ [\lambda q \in D_t\,.\ p = 1 \text{ or } q = 1]$

Exercise 2.9 (Chapter 2, p.40)

Replace the "?" in each of the following statements (you may want to review definition (5) of section 2.3 before tackling this exercise):

a) $\big[\lambda f \in D_{\langle e,t\rangle}\,.\ [\lambda x \in D_e\,.\ f(x) = 1 \text{ and } x \text{ is grey}]\big] \in D_?$

b) $\big[\lambda f \in D_{\langle e,\langle e,t\rangle\rangle}\,.\ [\lambda x \in D_e\,.\ f(x)(\text{Ann}) = 1]\big] \in D_?$

c) $\big[\lambda y \in D_e\,.\ \big[\lambda f \in D_{\langle e,t\rangle}\,.\ [\lambda x \in D_e\,.\ f(x) = 1 \text{ and } x \text{ is in } y]\big]\big] \in D_?$

d) $\big[\lambda f \in D_{\langle e,t\rangle}\,.\ \text{there is some } x \in D_e \text{ such that } f(x) = 1\big] \in D_?$

e) $\big[\lambda f \in D_{\langle e,t\rangle}\,.\ \text{Mary}\big] \in D_?$

f) $\big[\lambda f \in D_{\langle e,t\rangle}\,.\ \big[\lambda g \in D_{\langle e,t\rangle}\,.\ \text{there is no } x \in D_e \text{ such that } f(x) = 1 \text{ and } g(x) = 1\big]\big] \in D_?$

Solution 2.9

a) $D_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$
b) $D_{\langle\langle e,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle}$
c) $D_{\langle e,\langle\langle e,t\rangle\rangle,\langle e,t\rangle\rangle\rangle}$
d) $D_{\langle\langle e,t\rangle,t\rangle}$
e) $D_{\langle\langle e,t\rangle,e\rangle}$
f) $D_{\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle}$

4. More of English: Non-verbal Predicates, Modifiers, Definite Descriptions

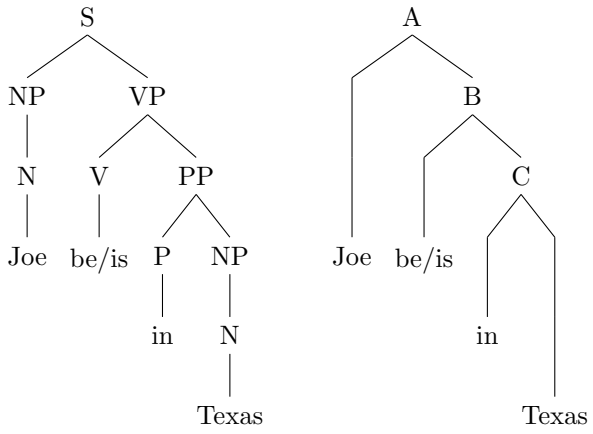
Exercise 4.1 (Chapter 4, p.63)
Calculate the truth-conditions for at least one of the sentences *Joe is in Texas*, *Joe is fond of Kaline*, and *Kaline is a cat*.

Solution 4.1
Before we start to calculate the truth conditions of the relevant sentences, one preliminary remark. Recall that in §4.1 Heim and Kratzer mention two strategies for how to incorporate 'semantically vacuous words' such as the preposition *of* (as it occurs in *fond of Kaline*) or the copula *be* (as it occurs in *is a cat*) into the semantics. The first way is to provide these expressions with their own lexical entries which assign them identity functions of the appropriate type. The second way is to 'assume that the semantic component simply "doesn't see" such items' (p.62). For specificity, we will go for the first option in this solution.

Let us begin by listing the relevant lexical entries. Since we want to cover all three of the sample sentences, the relevant list is the following (note that we additionally include a lexical entry for *gray* which we will make use of in exercise 4.2):

a) $[\![\text{Joe}]\!] = \text{Joe}$
b) $[\![\text{Kaline}]\!] = \text{Kaline}$
c) $[\![\text{Texas}]\!] = \text{Texas}$
d) $[\![\text{of}]\!] = \lambda x \in D_e \,.\, x$
e) $[\![\text{be}]\!] = \lambda f \in D_{\langle e,t\rangle} \,.\, f$
f) $[\![\text{a}]\!] = \lambda f \in D_{\langle e,t\rangle} \,.\, f$
g) $[\![\text{in}]\!] = \lambda x \in D_e \,.\, [\lambda y \in D_e \,.\, y \text{ is in } x]$
h) $[\![\text{cat}]\!] = \lambda x \in D_e \,.\, x \text{ is a cat}$
i) $[\![\text{fond}]\!] = \lambda x \in D_e \,.\, [\lambda y \in D_e \,.\, y \text{ is fond of } x]$
j) $[\![\text{gray}]\!] = \lambda x \in D_e \,.\, x \text{ is gray}$

Next let us give a derivation for *Joe is in Texas*.[7] There are several equivalent routes by which we can calculate the truth-conditions of this sentence. However, we always need to start from an assumption about what syntactical structure the sentence in question has. In fact, the semantics that Heim and Kratzer develop is fairly neutral with respect to questions of syntactic structure. For the purposes of calculating the truth-condition of a sentence we can ignore non-branching nodes since these simply inherit their denotation from their daughter node (see rule (2) *Non-Branching Nodes* on p.44). And we can also ignore the question of what syntactic categories individual nodes instantiate, since we only have one rule for deriving the denotation of a branching node, and this rule is blind to syntactic categories: *Functional Application* (see p.44). Thus, all we need to know is what the branching nodes are in a given sentence. Accordingly, while a possible syntactic analysis of *Joe is in Texas* is given by the following tree on the left hand side (including information about syntactic categories and non-branching nodes), we can make do with the less informative tree on the right hand side which simply identifies the branching nodes:



[7] Note that *is* as it occurs in *Joe is in Texas* is simply the third person singular form of the verb *be*. For the purposes of calculating truth conditions, we will not distinguish between the different forms of the verb *be*.

We can now calculate the truth-condition of the sentence. We will do this by calculating the denotations for all the branching nodes, starting with the lowest branching node (in the above trees: PP and C respectively, corresponding to the string *in Texas*), then the VP/B node (corresponding to *is in Texas*), and finally the S/A node (corresponding to the target sentence):[8]

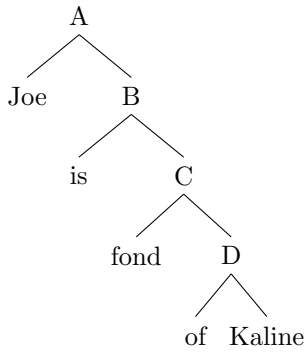
(1) By FA: $[\![\text{in Texas}]\!]$ $= [\![\text{in}]\!]([\![\text{Texas}]\!])$
(2) LE c) and g): $= [\lambda x \in D_e \,.\, [\lambda y \in D_e \,.\, y \text{ is in } x]]\,(\text{Texas})$
(3) λ-reduction: $= \lambda y \in D_e \,.\, y$ is in Texas

(4) By FA: $[\![\text{is in Texas}]\!]$ $= [\![\text{is}]\!]\,([\![\text{in Texas}]\!])$
(5) LE e), (3): $= \left[\lambda f \in D_{\langle e,t\rangle} \,.\, f\right](\lambda y \in D_e \,.\, y \text{ is in Texas})$
(6) λ-reduction: $= \lambda y \in D_e \,.\, y$ is in Texas

(7) By FA: $[\![\text{Joe is in Texas}]\!]$ $= [\![\text{is in Texas}]\!]([\![\text{Joe}]\!])$
(8) LE a), (6): $= [\lambda y \in D_e \,.\, y \text{ is in Texas}]\,(\text{Joe})$
(9) λ-reduction, Truth: $= 1$ iff Joe is in Texas.

Note that while in all of the lines (1), (4), and (7) same rule is being applied to a given branching node—namely *F*unctional *A*pplication—sometimes this rule will license the application of the denotation of the left daughter to the right daughter (as in lines (1) and (4)), while sometimes this order will be reversed (as in line (7)). The order will be determined by the type of the relevant denotations.

Next up *Joe is fond of Kaline*. Let us assume the following branching structure:



We can now proceed to calculate the truth conditions of the sentence:

(1) By FA: $[\![\text{of Kaline}]\!]$ $= [\![\text{of}]\!]([\![\text{Kaline}]\!])$
(2) LE b) and d): $= [\lambda x \in D_e \,.\, x]\,(\text{Kaline})$
(3) λ-reduction: $=$ Kaline

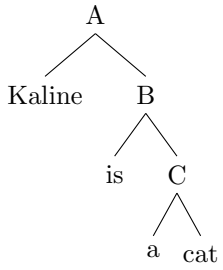
(4) By FA: $[\![\text{fond of Kaline}]\!]$ $= [\![\text{fond}]\!]\,([\![\text{of Kaline}]\!])$
(5) LE i), (3): $= [\lambda x \in D_e \,.\, [\lambda y \in D_e \,.\, y \text{ is fond of } x]]\,(\text{Kaline})$
(6) λ-reduction: $= \lambda y \in D_e \,.\, y$ is fond of Kaline

(7) By FA: $[\![\text{is fond of Kaline}]\!]$ $= [\![\text{is}]\!]([\![\text{fond of Kaline}]\!])$
(8) LE d), (6): $= \left[\lambda f \in D_{\langle e,t\rangle} \,.\, f\right](\lambda y \in D_e \,.\, y \text{ is fond of Kaline})$
(9) λ-reduction: $= \lambda y \in D_e \,.\, y$ is fond of Kaline

(10) By FA: $[\![\text{Joe is fond of Kaline}]\!]$ $= [\![\text{is fond of Kaline}]\!]([\![\text{Joe}]\!])$
(11) LE a), (6): $= [\lambda y \in D_e \,.\, y \text{ is fond of Kaline}]\,(\text{Joe})$
(12) λ-reduction, Truth: $= 1$ iff Joe is fond of Kaline.

Finally, let us turn to *Kaline is a cat*, which we will assume has the following branching structure:

[8] In the following semi-formal derivations, we use 'LE a)', 'LE b)' etc. to refer to the relevant lexical entries from the lexicon given above.



Here is the derivation:

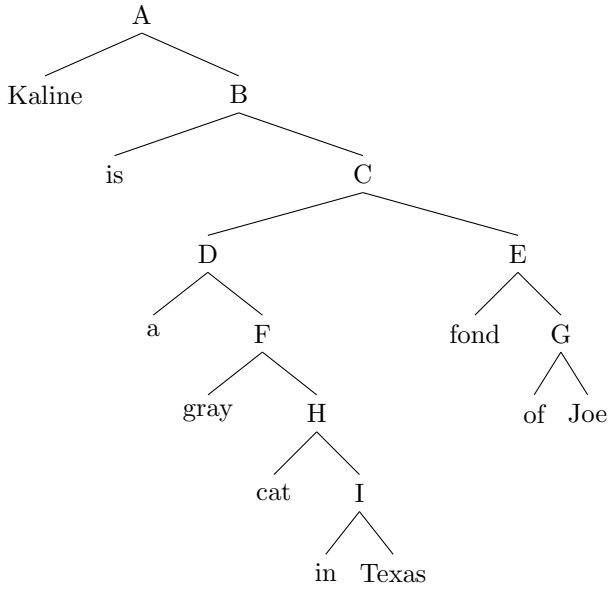
(1) By FA: $[\![\text{a cat}]\!]$ $= [\![\text{a}]\!]([\![\text{cat}]\!])$
(2) LE f) and h): $= \left[\lambda f \in D_{\langle e,t\rangle} \,.\, f\right](\lambda x \in D \,.\, x \text{ is a cat})$
(3) λ-reduction: $= \lambda x \in D_e \,.\, x \text{ is a cat}$

(4) By FA: $[\![\text{is a cat}]\!]$ $= [\![\text{is}]\!]\,([\![\text{a cat}]\!])$
(5) LE e), (3): $= \left[\lambda f \in D_{\langle e,t\rangle} \,.\, f\right](\lambda x \in D_e \,.\, x \text{ is a cat})$
(6) λ-reduction: $= \lambda x \in D_e \,.\, x \text{ is a cat}$

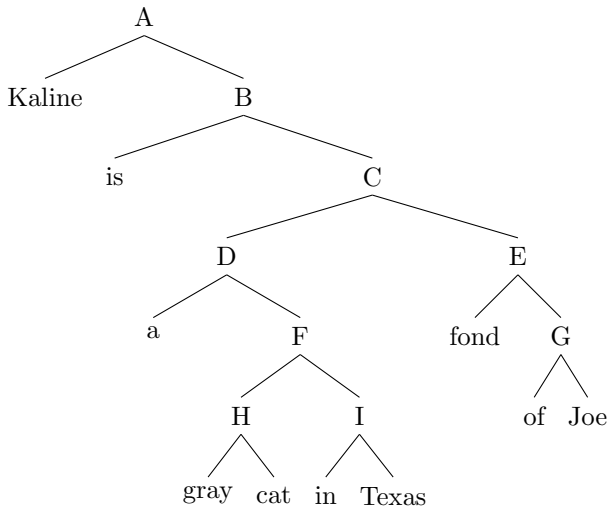
(7) By FA: $[\![\text{Kaline is a cat}]\!]$ $= [\![\text{is a cat}]\!]([\![\text{Kaline}]\!])$
(8) LE b), (6): $= [\lambda x \in D_e \,.\, x \text{ is a cat}]\,(\text{Kaline})$
(9) λ-reduction, Truth: $= 1$ iff Kaline is a cat.

Exercise 4.2 (Chapter 4, p.66)
Calculate the truth-conditions for (8) *Kaline is a gray cat in Texas fond of Joe*, given one possible syntactic parse.

Solution 4.2
With respect to examples such as *Kaline is a gray cat in Texas fond of Joe*, we can assign several distinct branching structures which would allow us to calculate a denotation of the sentence from the denotations of the lexical items via our rules (in particular via the rules *Functional Application* given on p.44 and *Predicate Modification*, given on p. 65). One possible branching structure is the following:



A second possible branching structure is given below:



Note the difference between these two branching structures: according to the first, *gray* combines with *cat in Texas*, whereas according to the second, *gray* combines with *cat* (and the resulting *gray cat* then combines with *in Texas*). However, in this case, the structural difference does not have semantic consequences at the sentential level; in both cases, we will derive the same truth-conditions. We will only give a derivation for the first branching structure (the derivation for the second branching structure produces the same truth-condition, albeit via a different route):

(1) FA: $[\![\text{of Joe}]\!] = [\![\text{of}]\!]([\![\text{Joe}]\!])$
(2) LE d) and a): $= [\lambda x\,.\,x]\,(\text{Joe})$
(3) λ-reduction: $= \text{Joe}$

(4) FA: $[\![\text{fond of Joe}]\!] = [\![\text{fond}]\!]\,([\![\text{of Joe}]\!])$
(5) LE i), (3): $= [\lambda x\,.\,[\lambda y\,.\,y \text{ is fond of } x]]\,(\text{Joe})$
(6) λ-reduction: $= \lambda y\,.\,y \text{ is fond of Joe}$

(7) FA: $[\![\text{in Texas}]\!] = [\![\text{in}]\!]([\![\text{Texas}]\!])$
(8) LE g) and c): $= [\lambda x\,.\,[\lambda y\,.\,y \text{ is in } x]]\,(\text{Texas})$
(9) λ-reduction: $= \lambda y\,.\,y \text{ is in Texas}$

(10) PM: $[\![\text{cat in Texas}]\!] = [\lambda x\,.\,[\![\text{cat}]\!](x) = [\![\text{in Texas}]\!](x) = 1]$
(11) LE h), (9) $= [\lambda x\,.\,[\lambda y\,.\,y \text{ is a cat}]\,(x) = [\lambda y\,.\,y \text{ is in Texas}]\,(x) = 1]$
(12) λ-reduction, Truth: $= \lambda x\,.\,x \text{ is a cat} \wedge x \text{ is in Texas}$

(13) PM: $[\![\text{gray cat in Texas}]\!] = [\lambda x\,.\,[\![\text{gray}]\!](x) = [\![\text{cat in Texas}]\!](x) = 1]$
(14) LE j), (12) $= [\lambda x\,.\,[\lambda y\,.\,y \text{ is gray}]\,(x) = [\lambda y\,.\,y \text{ is a cat} \wedge y \text{ is in Texas}]\,(x) = 1]$
(15) λ-reduction, Truth: $= \lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas}$

(16) FA: $[\![\text{a gray cat in Texas}]\!] = [\![\text{a}]\!]\,([\![\text{a gray cat in Texas}]\!])$
(17) LE f), (15): $= \left[\lambda f \in D_{\langle e,t\rangle}\,.\,f\right](\lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas})$
(18) λ-reduction: $= \lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas}$

(19) PM: $[\![\text{a g c in T f of J}]\!] = [\lambda x\,.\,[\![\text{a g c in T}]\!](x) = [\![\text{f of J}]\!](x) = 1]$
(20) (18) (6) $= [\lambda x\,.\,[\lambda y\,.\,y \text{ is g} \wedge y \text{ is a c} \wedge y \text{ is in T}]\,(x) = [\lambda y\,.\,y \text{ is f of J}]\,(x) = 1]$
(21) λ-reduction: $= \lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas} \wedge x \text{ is fond of Joe}$

(22) FA: $[\![\text{is a g c in T f of J}]\!] = [\![\text{be}]\!]\,([\![\text{a g c in T f of J}]\!])$
(23) LE e), (21): $= \left[\lambda f \in D_{\langle e,t\rangle}\,.\,f\right](\lambda x\,.\,x \text{ is g} \wedge x \text{ is a c} \wedge x \text{ is in T} \wedge x \text{ is f of J})$
(24) λ-reduction: $= \lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas} \wedge x \text{ is fond of Joe}$

(25) FA: $[\![\text{K is a g c in T f o J}]\!] = [\![\text{is a g c in T f of J}]\!]\,([\![\text{K}]\!])$
(26) LE b), (24): $= [\lambda x\,.\,x \text{ is g} \wedge x \text{ is a c} \wedge x \text{ is in T} \wedge x \text{ is f of J}](\text{Kaline})$
(27) λ-reduction, Truth: $= 1 \text{ iff K is gray} \wedge \text{K is a cat} \wedge \text{K is in Texas} \wedge \text{K is fond of Joe}$

Exercise 4.3 (Chapter 4, p.67)

Calculate the truth-conditions for (8) *Kaline is a gray cat in Texas fond of Joe*, given one possible syntactic parse. This time, use FA instead of PM [and use the revised lexical entries given in §4.3.2].

Solution 4.3

We will assume the same syntactic parse as in exercise 4.2. The calculation in exercise 4.2 employed *PM* (predicate modification) at three points: in step (10), where we combined ⟦cat⟧ with ⟦in Texas⟧, in step (13), where we combined ⟦cat in Texas⟧ with ⟦gray⟧, and in step (19), where we combined ⟦a gray cat in Texas⟧ with ⟦fond of Joe⟧. In order to do away with predicate modification, we will need to modify some lexical entries. One possible modification is the following

g*) $[\![\text{in}]\!] = \lambda x \in D_e\,.\,\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y \in D_e\,.\,f(y) = 1 \wedge y \text{ is in } x]\big]$
i*) $[\![\text{fond}]\!] = \lambda x \in D_e\,.\,\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y \in D_e\,.\,f(y) = 1 \wedge y \text{ is fond of } x]\big]$
j*) $[\![\text{gray}]\!] = \lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda x \in D_e\,.\,f(x) = 1 \wedge x \text{ is gray}]$

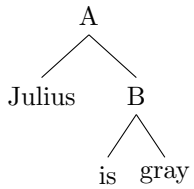
We can now give the revised derivation as follows:

(1) FA:	⟦of Joe⟧	= ⟦of⟧(⟦Joe⟧)
(2) LE d) and a):		= $[\lambda x\,.\,x]$(Joe)
(3) λ-reduction:		= Joe
(4) FA:	⟦fond of Joe⟧	= ⟦fond⟧ (⟦of Joe⟧)
(5) LE i*), (3):		= $\big[\lambda x\,.\,\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y\,.\,f(y) = 1 \wedge y \text{ is fond of } x]\big]\big]$(Joe)
(6) λ-reduction:		= $\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y\,.\,f(y) = 1 \wedge y \text{ is fond of Joe}]$
(7) FA:	⟦in Texas⟧	= ⟦in⟧(⟦Texas⟧)
(8) LE g*) and c):		= $\big[\lambda x\,.\,\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y\,.\,f(y) = 1 \wedge y \text{ is in } x]\big]\big]$(Texas)
(9) λ-reduction:		= $\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y\,.\,f(y) = 1 \wedge y \text{ is in Texas}]$
(10) FA:	⟦cat in Texas⟧	= ⟦in Texas⟧(⟦cat⟧)
(11) LE h), (9)		= $\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y\,.\,f(y) = 1 \wedge y \text{ is in Texas}]\big](\lambda x\,.\,x \text{ is a cat})$
(12) λ-reduction:		= $\lambda y\,.\,[\lambda x\,.\,x \text{ is a cat}](y) = 1 \wedge y \text{ is in Texas}$
(13) λ-reduction, Truth:		= $\lambda y\,.\,y \text{ is a cat} \wedge y \text{ is in Texas}$
(14) FA:	⟦gray cat in Texas⟧	= ⟦gray⟧(⟦cat in Texas⟧)
(15) LE j*), (13)		= $\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda x\,.\,f(x) = 1 \wedge x \text{ is gray}]\big](\lambda y\,.\,y \text{ is a c} \wedge y \text{ is in T})$
(16) λ-reduction:		= $\lambda x\,.\,[\lambda y\,.\,y \text{ is a c} \wedge y \text{ is in T}](x) = 1 \wedge x \text{ is gray}$
(17) λ-reduction, Truth:		= $\lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas}$
(18) FA:	⟦a gray cat in Texas⟧	= ⟦a⟧ (⟦gray cat in Texas⟧)
(19) LE f), (17):		= $\big[\lambda f \in D_{\langle e,t\rangle}\,.\,f\big](\lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas})$
(20) λ-reduction:		= $\lambda x\,.\,x \text{ is gray} \wedge x \text{ is a cat} \wedge x \text{ is in Texas}$
(21) FA:	⟦a g c in T f of J⟧	= ⟦fond of Joe⟧(⟦a gray cat in Texas⟧)
(22) (6) (20)		= $\big[\lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda y\,.\,f(y) = 1 \wedge y \text{ is f of J}]\big](\lambda x\,.\,x \text{ is g} \wedge x \text{ is a c} \wedge x \text{ is in T})$
(23) λ-reduction:		= $\lambda y\,.\,[\lambda x\,.\,x \text{ is g} \wedge x \text{ is a c} \wedge x \text{ is in T}](y) = 1 \wedge y \text{ is f of J}$
(24) λ-reduction, Truth:		= $\lambda y\,.\,y \text{ is gray} \wedge y \text{ is a cat} \wedge y \text{ is in Texas} \wedge y \text{ is fond of Joe}$
(25) FA:	⟦is a g c in T f of J⟧	= ⟦be⟧ (⟦a g c in T f of J⟧)
(26) LE e), (24):		= $\big[\lambda f \in D_{\langle e,t\rangle}\,.\,f\big](\lambda y\,.\,y \text{ is g} \wedge y \text{ is a c} \wedge y \text{ is in T} \wedge y \text{ is f of J})$
(27) λ-reduction:		= $\lambda y\,.\,y \text{ is gray} \wedge y \text{ is a cat} \wedge y \text{ is in Texas} \wedge y \text{ is fond of Joe}$
(28) FA:	⟦K is a g c in T f o J⟧	= ⟦is a g c in T f of J⟧ (⟦K⟧)
(29) LE b), (27):		= $[\lambda y\,.\,y \text{ is g} \wedge y \text{ is a c} \wedge y \text{ is in T} \wedge y \text{ is f of J}]$(Kaline)
(30) λ-reduction, Truth:		= 1 iff K is gray ∧ K is a cat ∧ K is in Texas ∧ K is fond of Joe

Exercise 4.4 (Chapter 4, p.67)

Define such a denotation. There are two distinct solutions.

More elaborate version of the exercise: Restrict your attention to *Julius is gray* (for the purposes of this question it raises precisely the same issues as *Julius is in Amherst*). Assume that *gray* has the lexical entry given in (11) on p.66. Accordingly, assume that ⟦gray⟧ is of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ (i.e. *gray* denotes a function from functions from objects to truth values to functions from objects to truth values). Moreover, assume that ⟦Julius⟧ is of type e, that ⟦Julius⟧ = Julius, and that the branching structure of *Julius is gray* is as follows:



We now want to calculate the truth-conditions of *Julius is gray* by appealing only to the rule of *Functional Application* (p.44). Define a denotation for *be/is* so that this can be done (i.e. define ⟦be⟧). Note that there are two possible answers. Provide one. (If you have time: provide both.)

Solution 4.4

Our goal is to calculate the truth-conditions of *Julius is gray*. We know the lexical entries for *Julius* and *gray*:

⟦Julius⟧ = Julius
⟦gray⟧ $= \lambda f \in D_{\langle e,t\rangle}\,.\,[\lambda x \in D_e\,.\,f(x) = 1$ and x is gray$]$

Our task is to define ⟦be⟧ in such a way that we can derive: ⟦Julius is gray⟧ = 1 iff Julius is gray. Since we know that ⟦Julius⟧ is of type e and that ⟦Julius is gray⟧ is of type t, we can infer that ⟦is gray⟧ must be of type $\langle e,t\rangle$: it must be a function from objects to truth values. We know that ⟦gray⟧ is of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. Hence, there are two semantic types that could be assigned to ⟦be⟧ so that ⟦be gray⟧ comes out as $\langle e,t\rangle$ according to *Functional Application*. *(i)* Either we assume that ⟦be⟧ is of type $\langle e,t\rangle$; in this case, *Functional Application* tells us that ⟦be gray⟧ = ⟦gray⟧(⟦be⟧). *(ii)* Or we assume that ⟦be⟧ is of type $\langle\langle\langle e,t\rangle,\langle e,t,\rangle\rangle,\langle e,t\rangle\rangle$; in this case, *Functional Application* tells us that ⟦be gray⟧ = ⟦be⟧(⟦gray⟧). On either option, we of course still have to decide on a *specific* denotation of the type in question.

Let us look at option *(i)* first. Which function of type $\langle e,t\rangle$ would ⟦be⟧ have to be in order to produce the correct result? In the end, we want the result that ⟦Julius is gray⟧ = 1 iff Julius is gray. Hence, we know that ⟦be gray⟧ = $[\lambda x \in D_e\,.\,x$ is gray$]$. Thus, we know that ⟦be⟧ must be a function $f \in D_{\langle e,t\rangle}$ such that ⟦gray⟧ maps f onto $[\lambda x \in D_e\,.\,x$ is gray$]$. But now note that ⟦gray⟧ will map f onto that function given that f maps every object onto 1. Hence, under the assumption that ⟦be⟧ is of type $\langle e,t\rangle$, it must be the constant function which maps every object to the truth value 1. One way of specifying this function is as follows:[9]

⟦be⟧ $= \lambda x \in D_e\,.\,x = x$

Let us turn to option *(ii)*. Which function of type $\langle\langle\langle e,t\rangle,\langle e,t,\rangle\rangle,\langle e,t\rangle\rangle$ would ⟦be⟧ have to be in order to produce the correct result? Again, we know that we need to end up with ⟦be gray⟧ = $\lambda x \in D_e\,.\,x$ is gray. However, this time we need ⟦be⟧ to map ⟦gray⟧ onto this function (rather than the other way around, as was the case on option *(i)*). Roughly speaking, ⟦gray⟧ takes a property F and produces another property, namely *is F and is gray*. That is precisely how we want the adjective *gray* to work as it occurs in constructions such as *gray dog*. However, in our target sentence *Julius is gray*, *gray* does not seem to have any property as input; we simply say that Julius is gray, not that he is a gray dog, or a gray man, or anything like that. So we have to perform a little trick. We will simply assume that *be/is* as it occurs in *is gray* provides the property that *gray* needs. But which property is that? Well, it can't be the property of being a dog, or the property of being a man, or of being a number, or anything like that; after all, some gray things are neither dogs, nor men, nor numbers. What we need (just as in the case of option *(i)* above) is a *trivial* property; a property that any object whatsoever has. Let us take *self-identity*. If *be/is* could somehow supply the property of being self-identical as the input for *gray* in *is gray*, then we would get the result we want; *is gray* would then be computed as *is self-identical and is gray*, which is logically equivalent with *is gray*. Here is a lexical entry for *be* which achieves this:

[9] Note that, as usual, we could have used many different ways to specify this function. For instance, we could just as well have said that ⟦be⟧ = $[\lambda x \in D_e\,.\,x$ is a natural number or x is not a natural number$]$; or that ⟦be⟧ = $[\lambda x \in D_e\,.$ if it rains, then it rains$]$; or simply that ⟦be⟧ = $[\lambda x \in D_e\,.\,1]$. Moreover, note that, if we *only* cared about ⟦be gray⟧, we could also have simply said that ⟦be⟧ = $[\lambda x \in D_e\,.\,x$ is gray$]$. However, in contrast to the lexical entry given above, this would of course start to give the wrong results once we consider cases in which *be* combines with adjectives distinct from *gray*, such as *blonde*.

$[\![\text{be}]\!] = \lambda f \in D_{\langle\langle e,t\rangle,\langle e,t,\rangle\rangle} . \left[\lambda x \in D_e . f\big(\lambda y \in D_e . y = y\big)(x) = 1\right]$

In words: $[\![\text{be}]\!]$ is a function which maps a function such as $[\![\text{gray}]\!]$ onto a function which maps an object x onto the truth value 1 iff applying $[\![\text{gray}]\!]$ first to the trivial function and then to x yields 1. We can make this a little easier on the eyes by introducing a symbol to denote the constant function which maps every object to the truth value 1. Lets call this function $\top$. We can now replace $\lambda y \in D_e . y = y$ with $\top$:

$[\![\text{be}]\!] = \lambda f \in D_{\langle\langle e,t\rangle,\langle e,t,\rangle\rangle} . [\lambda x \in D_e . f(\top)(x) = 1]$

Exercise 4.5 (Chapter 4, p.67)
It would not be adequate to list the two readings of each preposition or adjective separately in the lexicon, as if they had to be learned individually. Evidently, there is a systematic relation between the two readings, which makes one predicable, given the other. So we would want to list only one in the lexicon, and derive the other by means of a general *lexical rule*. Spell out two versions of this proposal. For the first version, assume that the homonyms with the simpler types ($\langle e,t\rangle$ or $\langle e,\langle e,t\rangle\rangle$) are basic and listed in the lexicon. For the second version, assume that the more complicated types $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ or $\langle e,\langle\langle e,t\rangle,\langle e,t\rangle\rangle\rangle$ are the ones of the basic, individually listed items. Your task is to formulate the appropriate lexical rules for either version. That is, you have to specify general recipes that map arbitrary denotations of the basic type to secondary denotations of the appropriate non-basic type. Rules of this kind are called *type-shifting rules*.

Solution 4.5
Heim and Kratzer have so far not been explicit about what precisely 'lexical rules' are supposed to be and what format they are supposed to have. For the following solution, I assume that we are supposed to give lexical rules in a format corresponding to the only explicitly given example for a lexical rule in Heim and Kratzer's book, the one on p.182. Hence, I will assume that lexical rules tell us that, given that our lexicon contains a lexical item δ_1 with a certain kind of denotation Δ_1, then our language also contains a related lexical item δ_2 with a related denotation Δ_2. In order to specify a lexical rule we will hence need to specify how we can obtain δ_2 from δ_1 and Δ_2 from Δ_1.

I will only give a solution for adjectives (*gray*, *loud*, *tall*); the application to other cases is rather straightforward. In the case of adjectives such as *gray*, our goal is to specify two type-shifting rules. The first takes a denotation of type $\langle e,t\rangle$ to a denotation of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$, and the second does the reverse; it takes a denotation of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ to a denotation of type $\langle e,t\rangle$. Let us start with the first type-shifting rule. In order to do so, we will define a function $\uparrow$. Under the assumption that $[\![\text{gray}]\!]$ is of type $\langle e,t\rangle$, we need $\uparrow$ to be behave in such a way that the following holds:

$[\![\text{gray}]\!] = \lambda x \in D_e . x \text{ is gray}$
$\uparrow([\![\text{gray}]\!]) = \lambda f \in D_{\langle e,t\rangle} . [\lambda x \in D_e . f(x) = 1 \text{ and } x \text{ is gray}]$

The following definition does the job:

$\uparrow = \lambda g \in D_{\langle e,t\rangle} . \left[\lambda f \in D_{\langle e,t\rangle} . [\lambda x \in D_e . f(x) = 1 \text{ and } g(x) = 1]\right]$

Thus, we can specify the first type-shifting rule as follows:

$\mathcal{R}_\uparrow$ For every adjective δ_1 with a meaning of type $\langle e,t\rangle$ there is a (homophonous and syntactically identical) item δ_2 with the following meaning of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$: $\uparrow[\![\delta_1]\!]$

Now let us assume that $[\![\text{gray}]\!]$ is of the more complex type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. In this case, we define a second function $\downarrow$, which we need to behave as follows:

$[\![\text{gray}]\!] = \lambda f \in D_{\langle e,t\rangle} . [\lambda x \in D_e . f(x) = 1 \text{ and } x \text{ is gray}]$
$\downarrow([\![\text{gray}]\!]) = \lambda x \in D_e . x \text{ is gray}$

The following definition does the job:[10]

$\downarrow = \lambda f \in D_{\langle\langle e,t\rangle,\langle e,t,\rangle\rangle} . [\lambda x \in D_e . f(\top)(x) = 1]$

[10] Recall from exercise 4.2 that I use $\top$ for the function of type $\langle e,t\rangle$ which maps every object to 1.

Thus, we can specify the second type-shifting rule as follows:[11]

$\mathcal{R}_{\downarrow}$ For every adjective δ_1 with a meaning of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ there is a (homophonous and syntactically identical) item δ_2 with the following meaning of type $\langle e,t\rangle$: $\downarrow[\![\delta_1]\!]$

[11] Note that $\downarrow$ is simply $[\![\text{be}]\!]$ as defined in the previous exercise. This makes sense: under the assumption that $[\![\text{gray}]\!]$ has the more complex type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$, we need $[\![\text{be}]\!]$ to be a type-shifter in order to treat sentences such as *Julius is gray*. Note also that $\downarrow$ simply reverses the effect of applying $\uparrow$: we have $\downarrow\uparrow(f) = f$.