Basic Categorial Semantics

Gary Hardegree
Department of Philosophy
University of Massachusetts
Amherst, MA 01003

1. Introduction

Having examined categorial syntax, we now turn our attention to categorial semantics. In this chapter, we examine Basic Categorial Semantics, which corresponds to Basic Categorial Syntax. In particular, we do not consider case-inflections or generalized-composition techniques.

2. The Basic Picture of Semantic Processing

We begin with the following **basic semantic flow-chart**.

step 1: **syntactic de-composition:**

a given phrase is broken down into its component phrases, and ultimately into its fundamental component phrases (morphemes).

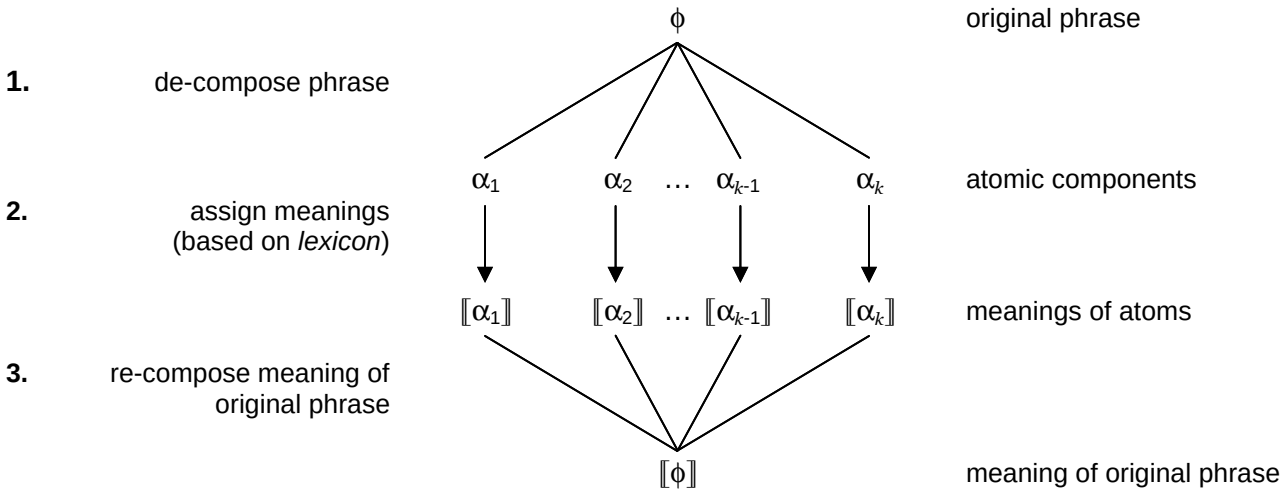
step 2: **lexical assignment of meanings:**

the fundamental component phrases are assigned *meanings* by consulting the *lexicon*.

step 3: **semantic re-composition:**

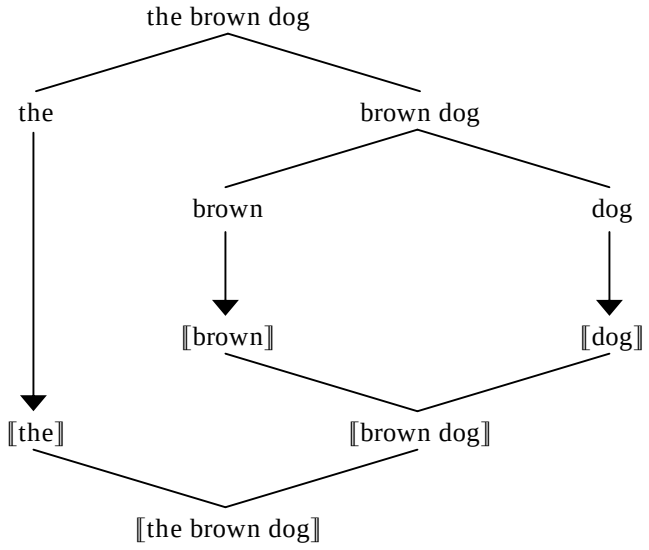
the meanings of the fundamental component phrases are combined to produce (*compute*) the meaning of the original phrase.

This process can be diagrammed as follows, where $[\![\alpha]\!]$ is the meaning of α.



3. An Example Phrase

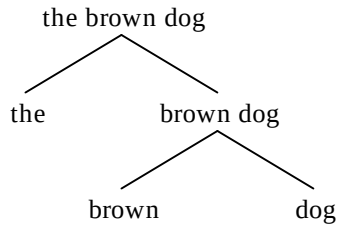
This diagram conceals important structural details in the de-composition and re-composition processes, since such details vary from phrase to phrase. By way of illustrating how the details might look, we consider a very simple example.



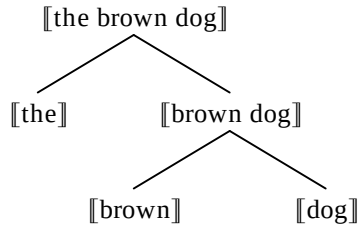
4. Isomorphism Thesis and Compositionality

Notice in the above diagram that **re**-composition (∨) formally *mirrors* **de**-composition (∧). We will generally find it convenient to draw everything "right-side-up", in which case we can depict these processes in parallel, as follows.

Syntactic De-Composition
(nodes are phrases)



Semantic Re-Composition
(nodes are meanings)



The two structures depicted above are *isomorphic* (i.e., structurally-identical). This observation is the basis of a fundamental thesis of categorial grammar – the **Isomorphism Thesis** – which may be summarized as follows.

> The semantics of a phrase is **isomorphic** to its syntax.

A key feature of the Isomorphism Thesis yields one the central tenets of modern formal semantics– the **Principle of (Local) Compositionality**, which may be stated as follows.

> The meaning of a compound phrase is composed out of
> the meanings of its **immediate constituents.**

5. Truth-Conditional Semantics – Denotations and Meanings

An outstanding question remains – what are meanings? We pursue a **truth-conditional model** of semantics, which may be described as follows.[1]

(1) the most basic units of significance are ***denotations*** (extensions);

(2) the **meaning** (intension) of a phrase α is what α denotes on each **occasion of usage**.

Let's discuss denotations (extensions) first. By way of illustration, consider the phrase 'the Eiffel Tower'; this phrase refers to a specific object – namely, the Eiffel Tower, which is the central architectural feature of Paris. We can formally describe this relation by saying either of the following.

the phrase 'the Eiffel Tower' **denotes** the Eiffel Tower
the denotation of the phrase 'the Eiffel Tower' is the Eiffel Tower

The phrase 'the Eiffel Tower' is an example of a phrase that has a **fixed-denotation**.[2] Other such phrases include logical terms – e.g., 'the', 'every', 'if' – as well as mathematical terms – e.g., 'two', 'seventy', 'plus'.

In contrast to these phrases, there are many phrases that have **variable-denotations**, or **occasion-dependent denotations**. For example, the phrase 'my dog' denotes different things according to who is speaking and when.[3] When *I* use the phrase 'my dog', it refers to *my* dog, insofar as I have a dog at the time of the utterance, but when *you* use this phrase it refers to *your* dog, insofar as you have a dog at the time of the utterance.[4]

Following Frege,[5] whereas the denotation of a proper-noun phrase is a particular individual,[6] the denotation of a declarative sentence is a truth-value (True or False).[7] As with proper-noun phrases, the denotation of a sentence may be context-dependent. For example, the sentence 'it is raining' has different denotations (i.e., truth-values) on different occasions.

We next turn to item (2). The central thesis of truth-conditional semantics is that the *meaning* of a sentence S is identified with the conditions under which S is true/false. More generally, the meaning of a phrase α is identified with the various denotations α has under the various conditions in which α may be uttered. We can describe this mathematically by saying that the meaning of phrase α is a *function* that takes each possible occasion of α-use (utterance) and yields what α denotes on that occasion.

---

[1] The truth-conditional model traces to Rudolf Carnap (1891-1970) [e.g., *Meaning and Necessity: a Study in Semantics and Modal Logic*, Chicago : University of Chicago Press, 1947], whose work is inspired by Gottlob Frege (1848-1925) [e.g., "Über Sinn und Bedeutung", *Zeitschrift für Philosophie und philosophische Kritik* 100 (1892): 25-50].

[2] This is an over-simplification, since replicas of the Eiffel Tower can appear in other places – for example, Las Vegas – in which case the expression 'the Eiffel Tower' can denote one of those objects. under the appropriate circumstances.

[3] Other contextual factors may arise. See Section 6.

[4] Even possession is context-dependent. For example, if I am hosting a dinner ask everyone to raise his/her glass for a toast, I am not asking each person to raise a glass he/she legally owns.

[5] See footnote 1. The translation of 'bedeutung' as 'denotation' is largely due to Alonzo Church, "A Formulation of the Logic of Sense and Denotation", in P. Henle et al. (eds), *Structure, Method and Meaning*, NY: Liberal Arts Press, 1951.

[6] For the sake of simplifying our account, we presume the proper-noun phrase in question is *singular*. More generally, a proper-noun phrase denotes an *entity*, which can be a *singular-entity*, a *plural-entity*, or a *mass-entity*. More about this later.

[7] We propose the capitalized words 'True' and 'False' to translate Frege's 'das Wahr' and 'das Falsche'. These, of course, are exactly the abstract objects encountered by elementary logic students in sentential logic.

6. What is a Situation?

We use the term 'situation' to refer to the formal-encoding of all the utterance-dependant factors that are relevant to computing the denotation of a phrase,[8] which include features of the utterance and features of the "world" independent of the utterance. At a minimum, a situation is expected to specify the following.

(1) the relevant *contextual* information, including

   a. the universe of discourse
   b. the denotations of all *indexical* expressions – 'I', 'you', 'now', 'here'
   c. the denotations of all *demonstrative* uses of words like 'this', 'that', 'he', 'she', and 'it'
   d. the denotations of all *ad hoc* proper-nouns

(2) the relevant *factual* information

Sometimes, the first items are collectively called a "context", and the second item is called a "possible world", and these can be profitably separated. For example, David Kaplan[9] proposes to distinguish *character* from *content*. A content is a function from possible worlds to denotations, and a character is a function from contexts to contents.[10]

7. Redrawing the Basic Picture of Semantic Processing

Now that we have both meanings (intensions) and denotations (extensions), we need to expand the basic picture of semantic processing, by adding the following step.

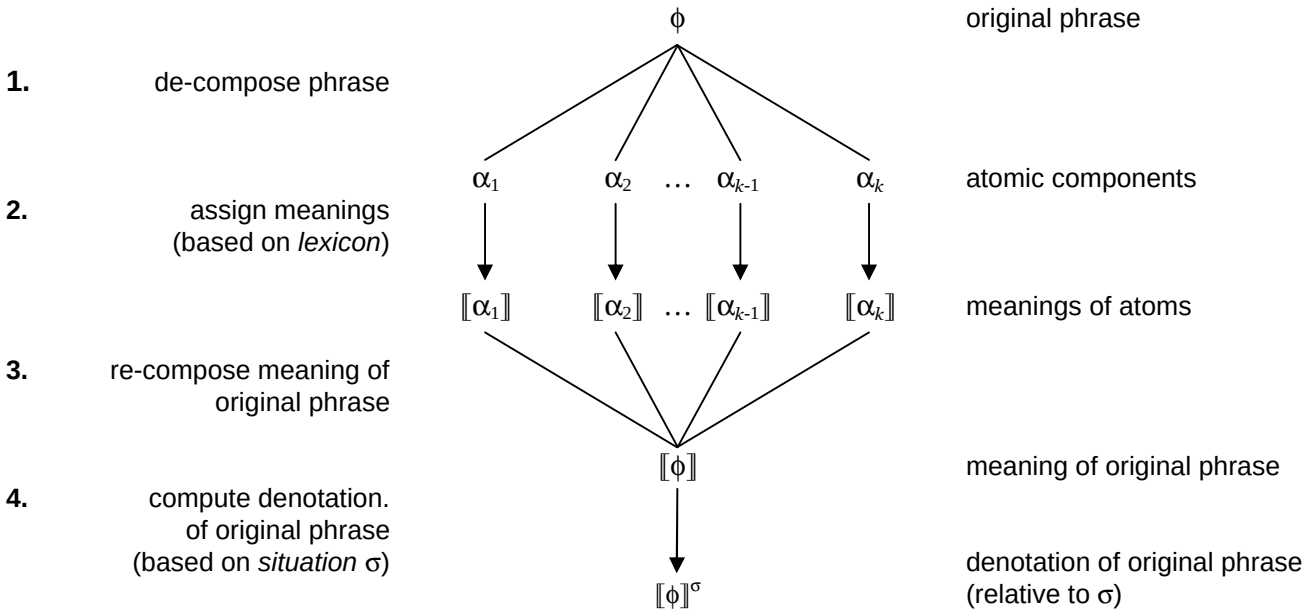
step 4: **denotation assignment:**
together with attendant conditions surrounding the utterance (i.e., the situation), the denotation of the original phrase is computed.

This expands our original diagram as follows.

[8] The calculation of a phrase's denotation is of course also dependent upon a prior assignment of lexical meanings and phrase-structure, but these factors are not situational.

[9] See, e.g., "Demonstratives: An Essay on the Semantics, Logic, Metaphysics, and Epistemology of Demonstratives and Other Indexicals", in *Themes From Kaplan*, Almog, J., Perry, J., and Wettstein, H. K., (eds), (New York: 1989), pp. 481 – 563

[10] Notice that, by categorial logic, Contexts→(Worlds→Denotations) ≡ (Contexts×Worlds)→Denotations.



In this diagram, we have the following identifications.

$[\![\alpha]\!]$ = the meaning/intension of α
$[\![\alpha]\!]^{\sigma}$ = the denotation/extension of α relative to σ

8. Extensional Semantics

The present work deals *primarily* with a very special type of semantics – *extensional* (or *denotational*) *semantics,* which is characterized as follows.

> In an extensional semantics, all semantic-composition is done,
> not with meanings/intensions, but with denotations/extensions.

Although this approach to semantics has serious shortcomings,[11] extensional semantics offers a very useful starting point for our investigations.

The following is the adjusted flow-chart for extensional semantics. Notice the reversal of steps three and four.

step 1: **syntactic de-composition:**
a given phrase is broken down into its component phrases, and ultimately into its fundamental (atomic) component phrases.

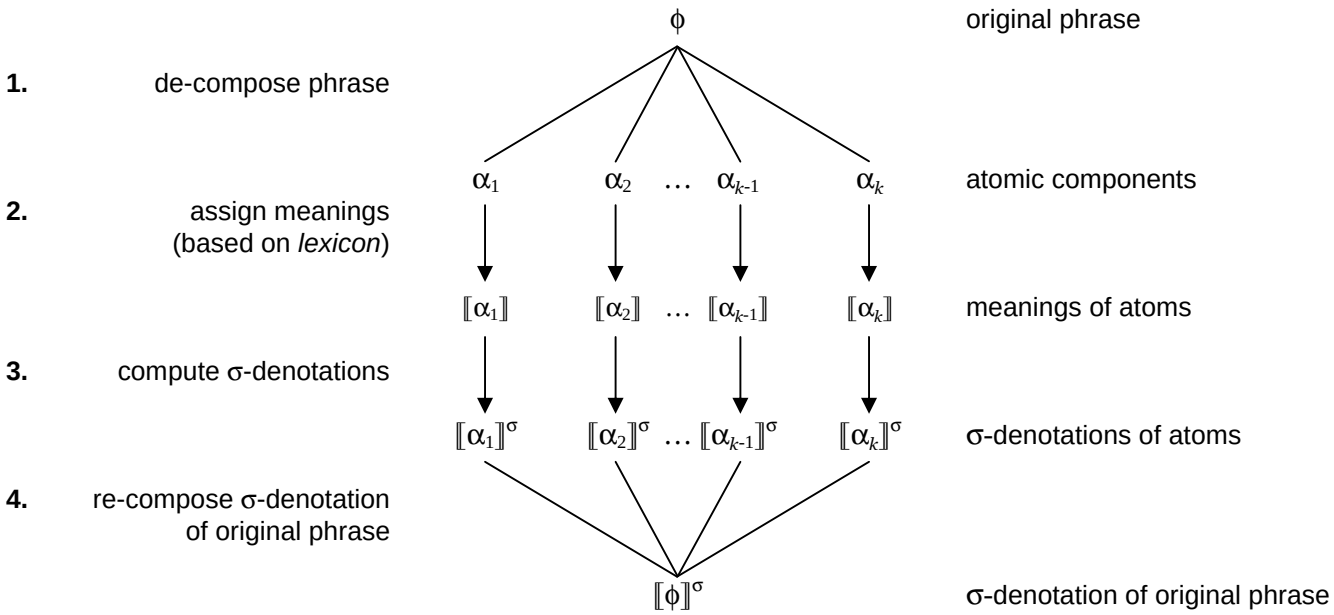
step 2: **lexical assignment of meanings:**
the fundamental component phrases are assigned *meanings* by consulting the *lexicon*.

step 3: **denotation assignment:**
the denotations of the fundamental components are computed by reference to the attendant contextual information [the *situation* σ];

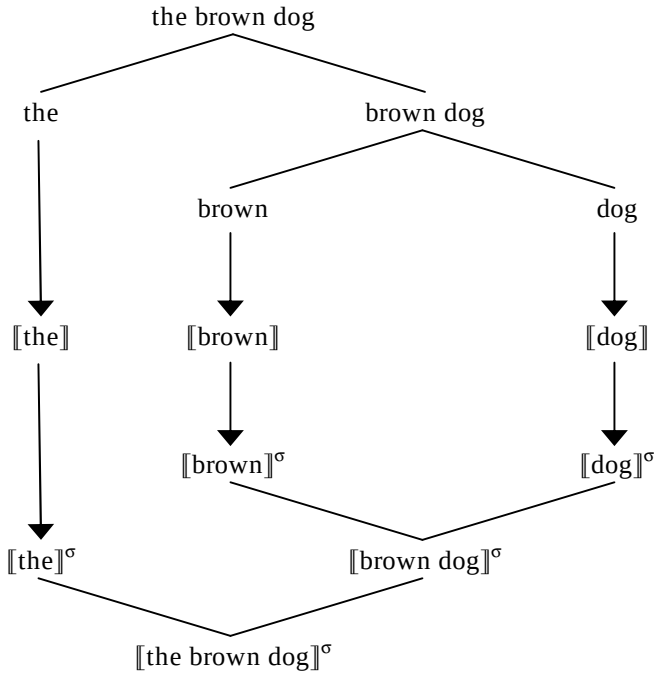
[11] The shortcomings pertain to modal, temporal, and other functors that act on intensions rather than extensions.

step 4: **semantic re-composition:**
the denotations of the fundamental component phrases are used to *compute* the denotation of the original phrase.

The following is the corresponding diagram. Here, by the σ-denotation of a phrase α, we mean the denotation of α relative to situation σ.



The following illustrates the process with our usual example.



9. Denotation Typology

Given the inductive character of syntactic-types, and given the isomorphism of syntax and semantics, we can construct a typology of denotations that exactly mimics the typology of phrases. By way of review, the syntactic-types are inductively constructed as follows.[12]

Primitive Syntactic-types

(1) D : proper-noun phrases
(2) S : sentences

Derivative Syntactic-types

(1) every primitive syntactic-type is a syntactic-type;
(2) if $\mathcal{A}$ and $\mathcal{B}$ are syntactic-types, then:
($\mathcal{A}\rightarrow\mathcal{B}$) is a syntactic-type;
($\mathcal{A}\times\mathcal{B}$) is a syntactic-type;
(3) nothing else is a syntactic-type.

In order to construct a parallel typology of denotations, we need merely identify the denotational counterparts of D and S, and then apply induction. We have already indicated the semantic counterparts of D and S. In particular, the denotation of a proper-noun phrase is an entity, and the denotation of a sentence is a truth-value (True or False). This yields the following initial semantic-types.

Primitive Semantic-types

(1) U : individuals (the universe/domain of discourse)
(2) V : truth-values {T,F}[13]

Once we have the primitive semantic-types, we can construct the full class of semantic-types by the following inductive construction.

Derivative Semantic-types

(1) every primitive semantic-type is a semantic-type;
(2) if $\mathcal{A}$ and $\mathcal{B}$ are semantic-types, then:
($\mathcal{A}\rightarrow\mathcal{B}$) is a semantic-type;
($\mathcal{A}\times\mathcal{B}$) is a semantic-type;
(3) nothing else is a semantic-type.

[12] At this point, for the sake of simplifying basic categorial semantics, and in anticipation of revised categorial semantics, we treat common-noun phrases derivative.

[13] We follow the usual custom in logic of using 'T' for the truth-value True and 'F' for the truth-value False. Note carefully however that, following Montague (see footnote 14), most linguists use the numerals '1' and '0' for this purpose. We find this mysterious, since it is patently obvious that truth-values are not numbers; for example, no one would seriously suggest that 'the number one' has the same denotation as 'snow is white'. At the same time, however, it is common practice to use numbers to *encode* abstract objects, and indeed we already follow this practice – in reference to cases, which we label using numerals.

10. Comparison with Montague's Typology

Richard Montague[14] also proposes a semantic typology, which is widely used in semantics, so it is useful to compare it to ours.

Montague proposes the following typology.

Primitive Montague Types

(1) e : entities
(2) t : truth-values

Based on these primitive types, he proposes the following inductive construction of the class of all semantic-types.

Derivative Montague Types

(1) every primitive type is a type;
(2) if $\mathcal{A}$ and C are types, then $\langle \mathcal{A}, C \rangle$ is a type;
(3) nothing else is a type.

The following are examples of Montague-types, together with their associated syntactic and semantic-types as we notate them.

Montague Type	Semantic-type	Syntactic-type
e	U	D
t	V	S
$\langle e, t \rangle$	U→V	D→S
$\langle e, \langle e, t \rangle \rangle$	U→(U→V)	D→(D→S)
$\langle \langle e, t \rangle t \rangle$	(U→V)→V	(D→S)→S

We use upper case letters in accordance with the usual syntactic practice of naming categories. We do not use the letters 'E' and 'T' for a number of reasons. We prefer to use 'T' for the truth value True, and we prefer to use 'E' for events, to be introduced later. We prefer arrow notation to ordered-pair notation because it is more suggestive of what the types do. This is explained further in the next section.

11. Types and their Associated Domains

Associated with each semantic-type, there is an associated domain, given by the following inductive definition.

Domains

(1) $dom(\mathsf{U})$ $=_{df}$ the underlying domain/universe of discourse
(2) $dom(\mathsf{V})$ $=_{df}$ the set {T,F} of truth-values
(3) $dom[\mathcal{A} \times \mathcal{B}]$ $=_{df}$ $dom(\mathcal{A}) \times dom(\mathcal{B})$
(4) $dom[\mathcal{A} \rightarrow \mathcal{B}]$ $=_{df}$ $dom(\mathcal{A}) \rightarrow dom(\mathcal{B})$

[14] Montague, R., "Universal Grammar", *Theoria*, 36 (1970):373-398.

On the right-hand side of the equations, the symbols '$\times$' and '$\rightarrow$' are used set-theoretically as follows.

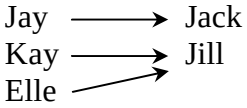
where A and B are sets,

$A \times B$ $=_{df}$ the Cartesian product of A and B
$A \rightarrow B$ $=_{df}$ the set of all functions from A into B

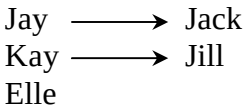
The Cartesian product of sets A and B is the set that contains all those ordered-pairs such that the first element is a member of A and the second element is a member of B.

We discuss functions in detail in the collateral chapter on set theory. Basically, a *function from A into B* is a (directed) **pairing** of elements of A with elements of B with the property that every element of A is paired with exactly one element of B.

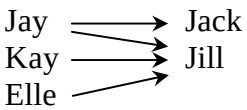
For example, the pairing



is a function from the set {Jay, Kay, Elle} into the set {Jack, Jill}. By contrast, the pairing



is not a function from the set {Jay, Kay, Elle} into the set {Jack, Jill}, since Elle is not paired with anyone[15]. Similarly, the following pairing



is not a function from {Jay, Kay, Elle} into {Jack, Jill}, since Jay is paired with two different individuals.[16]

12. Types = Associated Domains

In what follows, we propose simply to *identify* each semantic-type *with* its domain, in which case we have the following identities.

(1)	U	=	the underlying domain/universe of discourse
(2)	V	=	the set {T,F} of truth-values
(3)	$\mathcal{A} \rightarrow B$	=	the set of all functions from $\mathcal{A}$ into B
(4)	$\mathcal{A} \times \mathcal{B}$	=	the Cartesian product of $\mathcal{A}$ and B

[15] We presume that Elle $\neq$ Jay and Elle $\neq$ Kay.
[16] We presume that Jack $\neq$ Jill

13. The Type-Correspondence Principle

We have already suggested that there is a natural correspondence between syntactic-types and semantic-types. We now make that correspondence formal and official. In particular, we have the following **Type-Correspondence Rule**.

(1)	$[\![\mathsf{D}]\!]$	=	U
(2)	$[\![\mathsf{S}]\!]$	=	V
(3)	$[\![\mathcal{A}\times\mathcal{B}]\!]$	=	$[\![\mathcal{A}]\!]\times[\![\mathcal{B}]\!]$
(4)	$[\![\mathcal{A}\rightarrow\mathcal{B}]\!]$	=	$[\![\mathcal{A}]\!]\rightarrow[\![\mathcal{B}]\!]$
	$[\![\mathfrak{I}]\!]$	$=_{df}$	the semantic-type corresponding to syntactic-type $\mathfrak{I}$.

Examples

Syntactic-Type	Semantic-Type
D	U
S	V
(S×S)→S	(V×V)→V
D→S	U→V
D→(D→D)	U→(U→U)
D→(D→S)	U→(U→V)
(D→S)→S	(U→V)→V

A correspondence between syntactic-types and semantic-types also constrains the class of admissible semantic evaluations, which is codified by the following **Type-Correspondence Principle**.

type($[\![\alpha]\!]^{\sigma}$) = $[\![\text{type}(\alpha)]\!]$
equivalently: if type(α) = $\mathfrak{I}$, then type($[\![\alpha]\!]^{\sigma}$) = $[\![\mathfrak{I}]\!]$

Here, $[\![\alpha]\!]^{\sigma}$ is the denotation of α relative to situation σ, which we propose to omit when it is understood, and $[\![\mathfrak{I}]\!]$ is the semantic-type associated with syntactic-type $\mathfrak{I}$, which is provided by the Type-Correspondence Rule.

14. Simple Examples

In order to see how the categorial constraints work, let us consider a few examples.

1. One-Place Connectives

Our working hypothesis is that:

type(not) = S→S

So, by the type-correspondence principle:

$$\begin{aligned} \text{type}(\llbracket \text{not} \rrbracket) &= \llbracket S \to S \rrbracket \\ &= V \to V \\ &= \{T,F\} \to \{T,F\} \end{aligned}$$

In other words, 'not' denotes a function from {T,F} into {T,F}, which is to say a *one-place truth-function*.

Now, there are exactly four one-place truth-functions, which are listed as follows.[17]

(1) { T→T, F→T }
(2) { T→T, F→F }
(3) { T→F, F→T }
(4) { T→F, F→F }

Function (3), which maps T to F , and F to T, is of course the truth-function associated with 'not'. In other words,

$$\llbracket \text{not} \rrbracket = \{ T \to F, F \to T \}$$

2. Two-Place Connectives

Elementary logic presents other truth-functional connectives, the simplest of which is 'and', which may be categorized as follows.

$$\text{type}(\text{and}) = (S \times S) \to S$$

So, according to the type-correspondence principle,

$$\begin{aligned} \text{type}(\llbracket \text{and} \rrbracket) &= \llbracket (S \times S) \to S \rrbracket \\ &= (V \times V) \to V \\ &= (\{T,F\} \times \{T,F\}) \to \{T,F\} \end{aligned}$$

In other words, a two-place (truth-functional) connective denotes a two-place truth-function. Now, there are exactly sixteen two-place truth-functions, including the following.[18]

$$\{ T \times T \to T, T \times F \to F, F \times T \to F, F \times F \to F \}$$

The reader should verify that this is precisely the truth-function that is customarily associated with 'and'; for example, if both input truth-values are T, then the output truth-value is T; otherwise, the output truth-value is F.

[17] We propose a secondary use of the category-arrow to notate ordered-pairs; in particular, on this usage, $\alpha \to \beta$ is the ordered-pair consisting of α and β (in that order). The symbol is accordingly multiply-ambiguous, so context will be crucial to reading it.

[18] I propose to use the small cross '×' to denote the ordered-pair operator. $\alpha \times \beta$ is the ordered-pair consisting of α and β in that order.

3. One-Place Predicates

After connectives, the next simplest logical functors are predicates, the simplest of which are one-place predicates, which have type $D \to S$. For example,

$$\text{type(is a woman)} = D \to S$$

So, according to the type-correspondence principle, we have the following.

$$\begin{aligned} \text{type(} [\![\text{is a woman}]\!] \text{)} &= [\![D \to S]\!] \\ &= U \to V \\ &= U \to \{T,F\} \end{aligned}$$

In other words, the denotation of a one-place predicate is a function from the set U of individuals into the set {T,F} of truth-values. As explained in the chapter on set theory, such a function is called a *characteristic function* on U, which corresponds to a subset of U. For example, the following function

Jay	→	F
Kay	→	T
Elle	→	T

is a characteristic function on {Jay, Kay, Elle} that corresponds to the subset {Kay, Elle}, which in this example is the subset of women in our tiny example-universe.

4. Two-Place Predicates and Transitive Verbs

According to logicians, a transitive verb, such as 'respects', is categorized as a two-place predicate as follows.

$$\text{type(respects)} = (D \times D) \to S$$

So, according to the type-correspondence principle, we have the following.

$$\begin{aligned} \text{type}([\![\text{respects}]\!]) &= (D \times D) \to S \\ &= (U \times U) \to V \\ &= (U \times U) \to \{T,F\} \end{aligned}$$

In other words, 'respects' denotes a function that takes a pair of elements of the universe of discourse and delivers a truth-value. For example, we can imagine a situation in which U = {Jay, Kay}, and the following is the denotation of 'respects'.

Jay × Jay	→	T
Jay × Kay	→	T
Kay × Jay	→	F
Kay × Kay	→	T

In other words, in the envisioned situation, Jay respects himself and Kay, Kay respects herself but not Jay.

According to linguists, a transitive verb, such as 'respects', is categorized as follows.

$$\text{type(respects)} \quad = \quad D \rightarrow (D \rightarrow S)$$

In this case, according to the type-correspondence principle, we have the following.

$$\begin{aligned} \text{type}(\llbracket \text{respects} \rrbracket) &= D \rightarrow (D \rightarrow S) \\ &= U \rightarrow (U \rightarrow V) \\ &= U \rightarrow (U \rightarrow \{T,F\}) \end{aligned}$$

In other words, 'respects' denotes a function that take an individual in U and delivers a function, which in turn takes an individual in U and delivers a truth-value. The situation imagined above gets encoded by the following function.

Jay	→	Jay → T Kay → T
Kay	→	Jay → F Kay → T

5. Quantifier Phrases

As a final example, we consider the following higher-order type.

$$(D \rightarrow S) \rightarrow S$$

A functor of this type takes a one-place predicate as input, and delivers a sentence as output. The most common instances are quantifier phrases such as 'everyone', 'someone', and 'no one'.

According to the type-correspondence principle,

$$\llbracket (D \rightarrow S) \rightarrow S \rrbracket \quad = \quad (U \rightarrow V) \rightarrow V$$

Thus, a QP denotes a function that takes a characteristic function on U as input and delivers a truth-value as output. Alternatively speaking, a QP denotes a characteristic function on the set of characteristic functions on U. Alternatively speaking, a QP denotes a property of properties.

If we set $U = \{\text{Jay, Kay}\}$, then we have the following interpretations of 'every one', 'some one', and 'no one', respectively.

⟦every one⟧

Jay → T Kay → T	→	T
Jay → T Kay → F	→	F
Jay → F Kay → T	→	F
Jay → F Kay → F	→	F

⟦some one⟧

Jay → T Kay → T	→	T
Jay → T Kay → F	→	T
Jay → F Kay → T	→	T
Jay → F Kay → F	→	F

⟦no one⟧

Jay → T Kay → T	→	F
Jay → T Kay → F	→	F
Jay → F Kay → T	→	F
Jay → F Kay → F	→	T

15. Semantic Composition – Frege's Thesis

Recall the fundamental hypothesis of Basic Categorial Syntax.

all syntactic-composition is **functor-application**

In particular, every instance of syntactic-composition is an instance of applying a functor to one or more arguments.

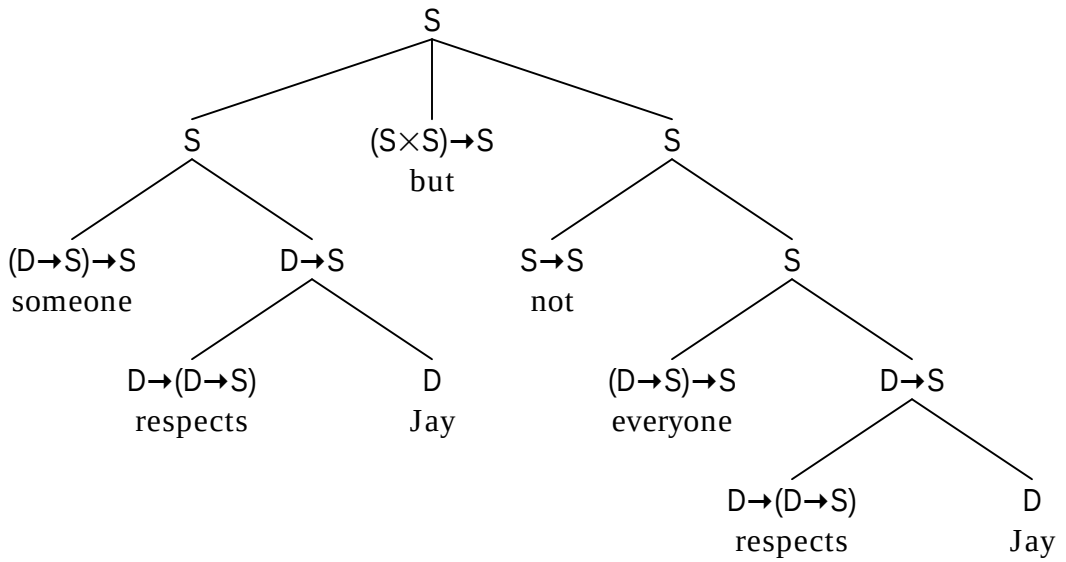
Now, we can combine this hypothesis with the isomorphism thesis, in which case we arrive at a very important principle of modern formal semantics, which traces to Gottlob Frege, and may accordingly be called *Frege's Thesis*.[19]

all semantic-composition is **function-application**

By way of illustration, we consider the following example.

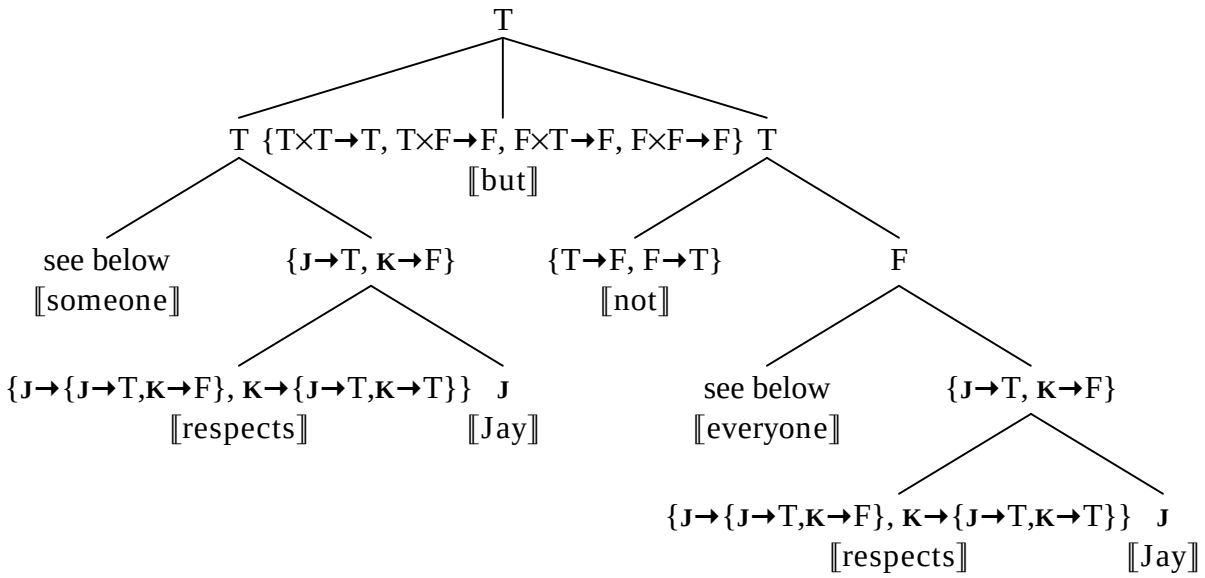
someone respects Jay but not everyone respects Jay

First, the syntactic-tree is given as follows.



The denotation (i.e., truth-value) of this sentence depends upon the situation in which it is uttered. By way of example, let us suppose that the "universe" consists of just Jay (**J**) and Kay (**K**), and suppose further that Jay respects himself and Kay, and Kay respects herself but not Jay. Then the corresponding semantic-tree looks thus.

[19] Since we abandon standard categorial syntax in favor of expanded categorial syntax, we will accordingly also abandon Fregean semantic-composition in favor of a considerably expanded account of semantic-composition.



⟦someone⟧ = { {J→T, K→T}→T, {J→T, K→F}→T, {J→F, K→T}→T, {J→F, K→F}→F }
⟦everyone⟧ = { {J→T, K→T}→T, {J→T, K→F}→F, {J→F, K→T}→F, {J→F, K→F}→F }

Note carefully that, in a semantic tree, the nodes are denotations of phrases. For example, the two bottom-most nodes include the denotation of 'Jay', which is the person Jay, and the denotation of 'respects', which is the function that (for example) maps Jay to the function that maps Jay to T and Kay to F.

By way of concluding this chapter, we observe that the notation we have adopted so far is unwieldy, even in such simple examples. For that reason, in the next chapter, we introduce a considerably more compact notation.