14 The Semantics of Classical First-Order Logic

1. Introduction

In order to describe semantic validity and entailment in the context of CFOL, we must provide the relevant class of admissible valuations. Obviously, since CFOL subsumes CSL, many of the features of CSL are preserved in the semantics of CFOL. In particular, the truth value of a sentential compound is truth-functionally related to the truth values of its parts. On the other hand, the truth value of a quantified formula is not so simply related to the truth value of the affiliated unquantified formula. For this reason, the semantics of CFOL is considerably more complex than the semantics of CSL.

2. Semantic Evaluations

The transition from CSL to CFOL involves a large increase in grammatical complexity. CSL has only one primitive grammatical category (sentences/formulas), and only one derivative grammatical category (connectives), so the semantics of CSL is quite simple. By contrast, CFOL has two primitive categories (sentences/formulas, nouns/singular terms), and three derivative categories (connectives, predicates, function signs), so the semantics is correspondingly more complex.

As mentioned before, at a minimum, a semantics specifies the class of (logically) admissible valuations on a formal language L, which are functions that assign truth values to every formula in L. In addition to its minimum duties, a semantics can be expected to provide a class of admissible semantic evaluations on L. A semantic evaluation is a function that assigns a semantic value $\upsilon(\varepsilon)$ to every grammatical expression ε, where $\upsilon(\varepsilon)$ is a semantic item that is appropriate to the grammatical category of ε[1].

3. Semantic Items and their Categories

Recall that every grammatical expression has a category. The primitive syntactic categories are N (noun phrase) and S (sentence); derivative syntactic categories include function signs (Nk→N), predicates (Nk→S), connectives (Sk→S).

Semantic categories parallel syntactic categories. First there are two primitive categories – individuals (U) , truth-values (V)[2]. Then the derivative semantic categories are constructed in a way that formally parallels the construction of the syntactic categories. The following are the relevant categories.

Category:	Instances:
U	elements of the universe U
V	elements of {T,V}; i.e., truth-values
(Uk→U)	k-place functions from U into U
(Uk→V)	k-place functions from U into {T,F}
(Vk→V)	k-place functions from {T,F} into {T,F}

[1] It is important to keep in mind the following point. Recall that, in CSL, an atomic formula is not true or false *per se*, but only relative to a particular valuation. The semantics of CSL specifies the admissible combinations of truth value assignments, but it does not specify the truth values of the atomic formulas. Similarly, the semantics of CFOL does not specify a unique denotation for each atomic symbol, but only specifies a class of admissible combinations of denotations. Accordingly, an atomic symbol of CFOL does not denote a specific item *per se*, but only relative to a given admissible semantic evaluation.

[2] If we are doing intensional logic, we must add other primitive semantic categories, including propositions and/or indices of various sorts, including worlds, times, etc.

Notice that we have already seen the category (Vk➔V); they are the truth-functions.

4. Official versus Conventional Identifications of Semantic Items

If we are pursuing a reductionistic set theory [see appendix on set theory], then we have an official formal account of semantic items. We briefly review that here.

First, a relation from A to B is a subset of A×B, which is the Cartesian product of A and B, which is the set {(a,b): a∈ A & b∈ B}; here, the ordered pair (a,b) is officially the unordered set {{a},{a,b}}. A function from A into B is a relation from A to B such that every element in A bears that relation to a unique element in B. A k-place function from A into B is a function from A^k into B. Here A^k is the k-fold Cartesian power of A, which is the set of all k-tuples of elements of A. A k-tuple of elements of A is officially a function from k into A. Generally A^B is the set of functions from B into A. The natural numbers are officially identified as follows. 0=∅; 1={∅}, 2={∅,{∅}}, etc. In other words, the natural number k is officially equal to the set of all its predecessors – 0=∅, 1={0}, 2={0,1}, etc.

It is usually convenient to make some conventional, or practical, identifications among the various set-theoretic entities built out of a set U of original elements.

(1) ordered pairs are identified with 2-tuples – (a,b)=⟨a,b⟩.

(2) the 1-tuple ⟨u⟩ is identified with u; accordingly, a 1-place function from U into U/V is the same as a function from U into U/V.

(3) we shift the indexing of every k-tuple, so that the first element of k-tuple (sequence) σ is σ_1, not σ_0. Note, officially $\sigma_i = \sigma(i)$.

(4) functions from U^k into V are identified with subsets of U^k. In particular, if S is a subset of U^k, $S\langle u_1,\ldots,u_k\rangle = T$ if $\langle u_1,\ldots,u_k\rangle \in S$; $S\langle u_1,\ldots,u_k\rangle = F$ if $\langle u_1,\ldots,u_k\rangle \notin S$. [See next section for a further account of this.]

(5) zero-place functions from U into U/V are identified with elements of U/V. [See next section for a further account of this.]

5. The Categorial Correspondence Rule

Basic to categorial formal semantics is the correspondence between syntactic categories and semantic categories. This correspondence is given by the following inductive rule.

N*	=	U
S*	=	V
$(\kappa_1,\ldots,\kappa_N\rightarrow\kappa_0)^*$	=	$(\kappa_1^*,\ldots,\kappa_N^*\rightarrow\kappa_0^*)$

First, N*=U means that the semantic counterpart, and hence the semantic value, of a noun phrase (N) is an element in the universe U of discourse (U). Similarly, the semantic counterpart of a sentence (S) is a truth-value (V).

The correspondence between derivative syntactic categories and derivative semantic categories is inductively generated. Let us do some simple examples. First, consider the category of one-place connectives – (S➔S). According to the correspondence rule,

$$(S \rightarrow S)^* \quad = \quad S^* \rightarrow S^*$$

But $S^* = V$, so

$$(S \rightarrow S)^* \quad = \quad S^* \rightarrow S^* = \quad V \rightarrow V$$

In other words, one-place connectives correspond to one-place truth-functions. This sounds about right – connectives denote truth-functions.

Similarly, consider the category of one-place function signs – $(N \rightarrow N)$. According to the correspondence rule,

$$(N \rightarrow N)^* \quad = \quad N^* \rightarrow N^*$$

But $N^* = U$, so

$$(N \rightarrow N)^* \quad = \quad N^* \rightarrow N^* \quad = \quad U \rightarrow U$$

In other words, one-place function signs correspond to one-place functions that assign elements of U to elements of U – U being the domain (universe) of individuals. This also sounds exactly right – function signs denote functions.

Finally, consider the category of one-place predicates – $(N \rightarrow S)$. According to the correspondence rule,

$$(N \rightarrow S)^* \quad = \quad N^* \rightarrow S^*$$

But $N^* = U$, and $S^* = V$, so

$$(N \rightarrow S)^* \quad = \quad N^* \rightarrow S^* \quad = \quad U \rightarrow V$$

In other words, one-place predicates correspond to one-place functions that assign truth-values to elements of the domain U of individuals. This does not sound *exactly* right! So let us see how it squares with our intuitive understanding of one-place predicates.

Intuitively, one place predicates correspond to properties; for example, 'is even' corresponds to the property of being even. Now, every property has an extension – associated with every property $\mathbb{P}$ is a subset $\{x: x \in U \ \& \ x \text{ has } \mathbb{P}\}$ of those individuals in U that have property $\mathbb{P}$, which is the extension of $\mathbb{P}$. Thus, property extensions are subsets of U.

Next, every subset S of U corresponds naturally to a function χ_S from U into {T,F}, defined as follows.

$$\begin{array}{lllll} \chi_S(e) = & T & \text{if} & e \in S; \\ = & F & \text{if} & e \notin S. \end{array}$$

The function χ_S is called the *characteristic function* of set S. [Usually, the characteristic function is defined using 1 in place of T, and 0 in place of F, but the idea is the same.]

Thus, we see that one-place predicates correspond to characteristic functions on U, which correspond to subsets of U, which correspond to properties of individuals in U.

Finally, we consider the degenerate case where k=0. What exactly is a zero-place function? What is A^0? Recall that, officially, $0=\varnothing$, so $A^0=A^{\varnothing}$. Officially, $A^{\varnothing}$ is the set of functions from $\varnothing$ into A; there is only one such object, the empty set $\varnothing$, so $A^{\varnothing} = \{\varnothing\}$. A function from $\{\varnothing\}$ into A can simply be identified with its value at $\varnothing$, so a zero-place function into U can be identified with an element of U, and a zero-place function into V can be identified with an element of V, a truth-value.

This is exactly how it should be. A zero-place function sign serves exactly like a proper noun; standing by itself, a zero-place function sign 'f' is a singular term. [Check the rules of formation!] Accordingly, its denotation should be an element of the universe U. Similarly, a zero-place predicate serves as a sentential constant; just like in SL, standing by itself, 'P' is a sentence; accordingly, its denotation should be an element of V, which is to say a truth-value.

6. The Categorial Composition Principle

Not only is there a correspondence between syntactic and semantic categories, this correspondence informs the interpretation of complex syntactic expressions. First of all, the interpretation of an expression of syntactic category K is a semantic item of the corresponding semantic category K*. In other words,

$$\text{cat}(\upsilon(\varepsilon)) = [\text{cat}(\varepsilon)]^*$$

Or:

if $\text{cat}(\varepsilon) = K$, then $\text{cat}(\upsilon(\varepsilon)) = K^*$

Furthermore, we have the following algebraic-compositional principle.

> Suppose ϕ is a syntactic functor, and $\varepsilon_1,\ldots,\varepsilon_k$ are appropriate syntactic arguments for ϕ, so that $\phi\langle\varepsilon_1,\ldots,\varepsilon_k\rangle$ is well-formed. Then for any semantic evaluation function υ:
>
> $$\upsilon(\phi\langle\varepsilon_1,\ldots,\varepsilon_k\rangle) = \upsilon(\phi)\langle\upsilon(\varepsilon_1),\ldots,\upsilon(\varepsilon_k)\rangle$$
>
> Alternatively:
>
> if:
> $\upsilon(\phi) = \phi^*$,
> $\upsilon(\varepsilon_1) = \varepsilon_1^*$,
> …,
> $\upsilon(\varepsilon_k) = \varepsilon_k^*$,
> then:
> $\upsilon[\phi\langle\varepsilon_1,\ldots,\varepsilon_k\rangle] = \phi^*\langle\varepsilon_1^*,\ldots,\varepsilon_k^*\rangle$

Here the angle brackets represent functor application on the left, and function application on the right.

We have already seen this principle in action in the context of classical sentential logic. Recall that an admissible valuation for CSL is any function υ satisfying the following requirements.

$\upsilon(\sim\alpha) = \sim\upsilon(\alpha)$
$\upsilon(\alpha\rightarrow\beta) = \upsilon(\alpha)\rightarrow\upsilon(\beta)$
etc.

Recall that we use the logical symbols ambiguously, for example, '$\sim$' refers both to syntactic and semantic conjunction, the latter being the truth-function that interprets the former. If we wanted to be more careful, we might rewrite this as follows.

$$\upsilon(\sim\alpha) = \sim^*\upsilon(\alpha)$$
$$\upsilon(\alpha\rightarrow\beta) = \upsilon(\alpha)\rightarrow^*\upsilon(\beta)$$
etc.

Here,

$\sim^*$ is the truth-function that interprets the connective $\sim$; i.e., $\sim^* = \upsilon(\sim)$
$\rightarrow^*$ is the truth-function that interprets the connective $\rightarrow$; i.e., $\rightarrow^* = \upsilon(\rightarrow)$
etc.

Writing these in generic categorial form, we have the following.

$$\upsilon(\sim\langle\alpha\rangle) = \upsilon(\sim)\langle\upsilon(\alpha)\rangle$$
$$\upsilon(\rightarrow\langle\alpha,\beta\rangle) = \upsilon(\rightarrow)\langle\upsilon(\alpha),\upsilon(\beta)\rangle$$
etc.

7. The Sub-Functions of a Semantic Evaluation

For the sake of delineating the different semantic duties of a semantic evaluation function, we sub-divide semantic evaluations into various inter-related sub-functions, described as follows. We begin the list with the overall category.

(0)	semantic evaluation	assigns denotations to all	grammatical expressions;
(1)	valuation	assigns denotations to all	formulas;
(2)	interpretation	assigns denotations to all	atomic non-logical symbols;
(3)	assignment function	assigns denotations to all	variables and constants;
(4)	designation function	assigns denotations to all	singular terms;
(5)	logical meaning function	assigns denotations to all	special logical symbols.

These are described in more detail in the following sections.

8. Interpretations

The first semantic evaluation functions we examine are called interpretations. An interpretation is a function that assigns a denotation to each non-logical atomic symbol; these include:

all proper nouns,
all functions signs,
all predicates except '='

Recall that '=' is a logical sign, so its denotation is provided by the logical meaning function.

The following is our official definition.

(D) Let L be a quantified language, let V be the set of truth-values, let U be a non-empty set, and let i be a function. Then i is an interpretation from L into U if and only if the following conditions are satisfied, for any non-logical symbol ε.

(1) if ε is a proper noun,
then $i(\varepsilon) \in U$;
(2) if ε is an n-place function sign,
then $i(\varepsilon)$ is an n-place function from U into U;
(3) if ε is an n-place non-logical predicate,
then $i(\varepsilon)$ is an n-place function from U into V;
(4) otherwise, $i(\varepsilon)$ is undefined.

(D) Let L be a quantified language, let i be a function. Then i is an interpretation on L if and only if there is a set U such that i is an interpretation from L into U.

The set U is called the domain (or universe) of discourse; it is what the quantifiers implicitly refer to; thus, '∀' is interpreted to mean 'every element of U is such that', and '∃' is interpreted to mean 'at least one element of U is such that'.

9. Truth and Falsity for Simple Atomic Formulas

Once we have an interpretation i on FOL L, we can immediately define the corresponding valuation function restricted to simple atomic formulas. These are officially defined as follows.

(D) Let L be a FOL, and let α be a formula in L. Then α is said to be a *simple atomic formula* if and only if α has the form $\mathbb{P}n_1...n_k$, where $\mathbb{P}$ is a k-place predicate, and $n_1,...,n_k$ are proper nouns.

Consider a simple atomic formula $\mathbb{P}n_1...n_k$. By general categorial principles, we require that:

$$\upsilon(\mathbb{P}n_1...n_k) = \upsilon(\mathbb{P})\langle\upsilon(n_1), \ldots, \upsilon(n_k)\rangle$$

Now, as we have mentioned, the semantic duties of υ are subcontracted to its sub-function i for $\mathbb{P}$, as well as $n_1, \ldots, n_k$; in other words,

$$\begin{aligned}
\upsilon(\mathbb{P}) &= i(\mathbb{P}) \\
\upsilon(n_1) &= i(n_1) \\
&\ldots \\
\upsilon(n_k) &= i(n_k).
\end{aligned}$$

So we have:

$$\upsilon(\mathbb{P}n_1...n_k) = i(\mathbb{P})\langle i(n_1), \ldots, i(n_k)\rangle$$

Or if we regard $i(\mathbb{P})$ as a subset of U, then we have:

$$\begin{aligned}
\upsilon(\mathbb{P}n_1...n_k) &= \text{T} \quad \text{if} \quad \langle i(n_1), \ldots, i(n_k)\rangle \in i(\mathbb{P}) \\
&= \text{F} \quad \text{otherwise}
\end{aligned}$$

The intuition is straightforward, even if the official formulation might seem somewhat opaque. According to interpretation i, the n-place predicate $\mathbb{P}$ denotes a certain n-place relation on U; call this

relation R. Also, according to i, each of proper nouns $n_1,...n_k$ denotes an object in U; call these objects $o_1,...,o_k$, respectively. In other words, we have the following.

$i(\mathbb{P}) = R$
$i(c_1) = o_1$
$i(c_2) = o_2$
etc.

Here, '$\mathbb{P}$' is the metalanguage name of the predicate, and 'R' is the metalanguage name of the relation denoted by that predicate, according to interpretation i. Similarly with the other expressions in the above list: expressions on the right refer to semantic items in U, whereas expressions on the left refer to syntactic items in L.

Now, the definition of truth for simple atomic formulas says that $\mathbb{P}n_1...n_k$ is true (relative to υ [i]) if and only if the tuple $\langle i(n_1),...i(n_k)\rangle$ is an element of i(P), but given the identities of i(P), etc., we have that $Pn_1...n_n$ is true in i if and only if the tuple $\langle o_1,...,o_k\rangle$ is an element of R.

Recall the set-theoretic definition of 'bears', first for 2-place relations, and generally for n-place relations.

(d) a bears R to b $=_{df}$ $\langle a,b\rangle \in R$
(d) $a_1,...,a_k$ bear R $=_{df}$ $\langle a_1,...,a_k\rangle \in R$

Thus, we have that $Pn_1...n_k$ is true in i if and only if the objects $o_1,...,o_k$ bear the relation R. In other words, we have the obvious result that a simple atomic formula $Pn_1...n_k$ is true in i if and only if the objects denoted by the proper nouns $n_1,...,n_k$ bear the relation denoted by the predicate P.

For the sake of further illustration, consider a simple example in the style of intro logic. In particular, consider the two-place predicate 'R', and the proper nouns 'j' and 'k'; also let U be the class of humans. Now, obviously, whether the formula 'Rjk' is true or not depends upon the answers to the following four questions.

(q1) what item does 'j' denote?
(q2) what item does 'k' denote?
(q3) what item does 'R' denote?
(q4) does the item in #1 bear the item in #3 to the item in #2?

The answers to these questions are as follows (where we drop quotes):

(a1) i(j)
(a2) i(k)
(a3) i(P)
(a4) yes, if i(j) bears i(R) to i(k); no, otherwise.

In other words, the formula 'Rjk' is true according to i if and only if i(j) bears i(R) to i(k).

For example, suppose the following.

'j' denotes Jay;	i.e.,	i(j) = Jay;
'k' denotes Kay;	i.e.,	i(k) = Kay;
'R' denotes the respects-relation;	i.e.,	i(R) ={$\langle$x,y$\rangle$: x respects y}

Then ‘Rjk’ is true in i if and only if
$\langle$Jay, Kay$\rangle \in \{\langle x,y\rangle$: x respects y$\}$,
which is true if and only if
Jay respects Kay.

10. Pronouns – Variables and Constants

As it stands, the semantic pattern described so far works perfectly for variable-free formulas, but it faces problems as soon as we consider variables. Variables are the logical counterparts of pronouns in natural language. Consider the following sentence.

he is right-handed.

In reference to this sentence, we can ask three related, but distinct, questions.

what does it mean?
what does it say?
is what it says true?

In particular, we can understand what it means, without knowing what it says; similarly, we can know what it says without knowing whether what it says is true.

The problem revolves around the pronoun ‘he’. If we do not know the occasion of its use, we do not know what ‘he’ refers to. Without this piece of information, we do not know what is being said, so (usually) we don’t know whether it is true. On the other hand, it is fairly clear that we understand what the sentence means.

Its meaning in fact contributes to the meanings of the following more complex sentences.

if Jay is not left-handed, then he is right-handed;
if a man is not left-handed, then he is right-handed.

Note carefully that there is an ambiguity in the word ‘he’. Either it takes an earlier noun phrase as antecedent, or it is used demonstratively (which requires an act of pointing while it is being uttered). We could convey this with a special pointer marker next to ‘he’, as follows.

if Jay is not left-handed, then he(☝) is right-handed;	[demonstrative]
if Jay is not left-handed, then he(☜) is right-handed;	[antecedent pointing]
if a man is not left-handed, then he(☝) is right-handed;	[demonstrative]
if a man is not left-handed, then he(☜) is right-handed;	[antecedent pointing]

The two demonstrative sentences may be translated into predicate logic as follows.

$\sim Lj \rightarrow Rx$
$\forall x\{Mx \rightarrow .\ \sim Lx \rightarrow Ry\}$

Notice that both sentences contain a free variable, which represents the demonstrative ‘he’. This is not obviously the best approach, however. Perhaps it is better to use constants (unquantified variables) to translate demonstrative pronouns, in which case we have the following translations.

$\sim Lj \rightarrow Rc$
$\forall x\{Mx \rightarrow . \sim Lx \rightarrow Rc\}$

In this case, the constant 'c' is used as a demonstrative pronoun – what we might also call an *ad hoc name*.

In the case of the other two sentences, we have the following translations.

$\sim Lj \rightarrow Rj$
$\forall x\{Mx \rightarrow . \sim Lx \rightarrow Rx\}$

Notice that these formulas have no free variables. Notice also how differently the two translations deal with the word 'he'. This reflects the important grammatical fact that there are two ways that a pronoun can be antecedent-pointing – either as a "pronoun of laziness", or as an *anaphoric* pronoun. In the first case, the pronoun may simply be replaced by its antecedent. In the second case, the pronoun cannot be replaced by its antecedent. The above examples illustrate this idea; so do the following.

Jay respects his(☜) mother	≡	Jay respects Jay's mother
every man respects his(☜) mother	≢	every man respects every man's mother

The basic idea about the semantics of logical variables/constants may be summarized as follows.

c	=:	it(☝)
x	=:	it(☜)

The actual colloquial reading will of course depend upon natural language grammatical conventions concerning gender (e.g., 'he' versus 'she' versus 'he/she') and case (e.g., 'he' versus 'him').

This assumes that we use constants exclusively as demonstratives. If we use free variables this way then we have a different semantic account. In particular, when a variable occurs free, it has a demonstrative (☝) interpretation, but when a variable occurs bound, it has an anaphoric (☜) interpretation. This means that

when 'x' is unbound,	'x'	means	'it(☝)'
when 'x' is bound,	'x'	means	'it(☜)'.

In either case, 'Fx' means 'it is F'.

11. Multiple Pronouns

Of course, the advantage of logical syntax over natural language syntax is that logical syntax has an infinite list of pronouns ('x', 'y', 'z', etc.), not just one ('it'+gender+case). This allows more complicated constructions – we can have 'the first *it*', 'the second *it*', etc. Also, each occurrence of 'it' can be demonstrative or anaphoric.

Let us do a couple of examples, assuming that constants are used exclusively as demonstrative pronouns, and variables are used as anaphoric pronouns. First, the formula

$\forall x\{Wx \rightarrow Rxc\}$

can be read:

every woman is such that she(☜) respects her(☝),

whose more colloquial form is:

every woman respects her(☝).

On the other hand, the formula

$\forall x\{Wx \rightarrow Rxy\}$

can be read:

every woman is such that she(☜) respects her(☜),

whose more colloquial form is:

every woman respects her(☜).

Here, it is understood that 'her' does not point at 'every woman', but to some earlier noun phrase. If we want 'her' to point at 'every woman, we write:

$\forall x\{Wx \rightarrow Rxx\}$;
every woman respects herself(☜).

[[Notice that reflexive pronouns (e.g., 'herself') are always antecedent-pointing, and never purely demonstrative.]]

Next, consider the formula,

$\exists y\{Wy \,\&\, \forall x\{Wx \rightarrow Rxy\}\}$,

which reads:

there is a woman such that every woman is such that she(☜) respects her(☜).

Even with the index fingers as written, the English sentence is ambiguous, because we do not know the respective antecedents of the two pronouns; which woman is which? That is why we need to further delineate our index fingers (pronouns). The following is the intended reading.

there is a $woman_1$ such that every $woman_2$ is such that she_2(☜) respects her_1(☜).

The same thing happens with demonstrative pronouns. Consider the army drill instructor selecting "volunteers" for the garbage detail.

I select you, you, you, and you.

Or, using 'him' instead of 'you'.

I select him, him, him, and him.

Presumably each utterance of 'him' is accompanied by an appropriate pointing gesture. Using delineated indexical markers, we can convey this as follows.

I select him_1(☝), him_2(☝), him_3(☝), and him_4(☝).

This suggests the following refinement of the semantics of variables/constants.

x =: $it_1(☜)$
y =: $it_2(☜)$
z =: $it_3(☜)$
etc.

a =: $it_1(☝)$
b =: $it_2(☝)$
c =: $it_3(☝)$
etc.

Or, if we wish to be more mathematically exact, we begin with an infinite sequence $\langle v_1,v_2,\ldots\rangle$ of variables, and an infinite sequence $\langle c_1,c_2,\ldots\rangle$ of constants, to which we propose the following reading.

v_k =: $it_k(☜)$
c_k =: $it_k(☝)$

12. Assignment Functions – Constants

In order to give an interpretation to all formulas at once, we must assign denotations to all constants (demonstrative pronouns) at once. Semantically this requires a universal act of pointing – What is $it_1(☝)$? What is $it_2(☝)$? etc. This may be mathematically accomplished in one of two ways. (1) We can specify an infinite sequence σ of elements of the domain U; thereby selecting σ_1 as the (demonstrative) denotation of c_1, σ_2 (demonstrative) as the denotation of c_2, etc. (2) Alternatively, we can use an assignment function, a, to assign a demonstrative denotation, $a(c)$, to every constant c. These are interchangeable. Given a sequence σ, we define a so that $a(c_k) = \sigma_k$. Given an assignment function, we define $\sigma = \langle a(c_1), a(c_2),\ldots\rangle$. The latter depends, of course, on having an enumeration $\langle c_1,c_2,\ldots\rangle$ of the constants.

Once we have chosen an object for each '$it_k(☝)$' to refer to, we can assign semantic values to sentences involving constants. For example, the sentence

$it_1(☝)$ is $\mathbb{P}$ $\quad\quad$ $[\mathbb{P}c_1]$

is true if and only if the object 'it_1' points at has the property that $\mathbb{P}$ denotes, which is to say:

$a(c_1) \in i(\mathbb{P})$,

or if we are using the functional guise of $\alpha(\mathbb{P})$,

$i(\mathbb{P})\langle a(c_1)\rangle = T$

13. Assignment Functions – Variables

But what about the "☜" pronouns? How do we assign a truth-value to:

$it_1(☜)$ is $\mathbb{P}$ $\quad\quad$ $[\mathbb{P}x_1]$

Without a linguistic context, we don't know what '$it_1(☜)$' points at. All we know is that it stands and waits (and therefore serves, if Milton is correct!) ready to point at an antecedent noun phrase, should such

an expression be placed grammatically "ahead" of it [sometimes, the antecedent is later in the actual sentence].

At this point, it seems that we have two choices.

(1) We can simply say that dangling pronouns (free variables) have no denotation, because they are semantically incomplete, and accordingly formulas with dangling pronouns (free variables) have no truth-value, because they are semantically incomplete.

(2) We can assign denotations to variables in a largely arbitrary manner.

Neither of these approaches is completely without inconvenience. We follow the latter approach, and assign denotations to variables in precisely the same way that we assign denotations to constants. This amounts to saying that we treat dangling pronouns as demonstrative pronouns.

This allows us to assign a truth-value to open formulas. For example, presuming that 'x' = v_1,

Fx [it_1(☝) is F]

is true if the first object σ_1 [i.e., the first "it" pointed at] has the property denoted by the predicate 'F'. Similarly, further presuming 'y' = v_2,

Fx & Fy [it_1(☝) is F and it_2(☝) is F]

is true if both object σ_1 [i.e., the first "it" pointed at] and object σ_2 [i.e., the second "it" pointed at] have the property denoted by predicate 'F'.

We now officially define assignment function.

(D) Let L be a quantified language, let Var(L) be the variables in L (i.e., Var(L) = {v_1, v_2,...}), let Con(L) be the constants in L (i.e., Con(L) = {c_1, c_2,...}), and let U be a non-empty set. Then an assignment function from L into U is any function from Var(L)∪Con(L) into U.

In other words, an assignment function assigns an object (in the domain) to each variable and constant.

14. Designation Functions – Singular Terms

Given an interpretation function i, and an assignment function a, from language L into domain U, we can define an associated designation function d from L into U as follows.

(D) Let L be a quantified language, let U be a non-empty set, let i be an interpretation from L into U, and let a be an assignment function from L into U. Then the designation function, d, associated with i and a is the function, d, satisfying the following conditions.

(1) the domain of d is set of all singular terms of L;
(2) if τ is a variable/constant, then $d(\tau)=a(\tau)$;
(3) if τ is a proper noun, then $d(\tau)=i(\tau)$;
(4) if τ is molecular, then τ is of the form $\phi\tau_1...\tau_n$, and $d(\tau) = i(\phi)\langle d(\tau_1),...,d(\tau_n)\rangle$.

In clause (4), the intuition is fairly simple. The denotation of a complex singular term is determined by the denotations of its various parts. Consider a simple example; a one-place function sign 'f', and a proper noun 'k'. Then the denotation of 'fk' will be determined by what 'f' and 'k' individually denote, which

are i(f) and d(k), respectively. The former is a one-place function; the latter is an individual in the domain. The denotation of 'fk' is obtained by applying the function i(f) to the individual d(k), which is to say

$$d(\text{fk}) = i(\text{f})\langle d(\text{k})\rangle,$$

or equivalently

> if 'k' denotes object o,
> and 'f' denotes function g,
> then 'fk' denotes object g(o).

A simple example shows how simple the intuition is. Suppose the following.

> 'k' denotes Kay; i.e., $d(\text{k})=i(\text{k})$ = Kay
> 'f' denotes the father-function; i.e., $i(\text{f}) = \{\langle x,y\rangle$: y is x's father$\}$;
>
> then 'fk' denotes Kay's father.

15. Truth-Valuations – Non-Quantified Formulas

In the present section, we give an account of truth-values for all non-quantified formulas of the language of CFOL. This is accomplished by first defining truth-value for atomic formulas, then inductively extending it to SL-molecular formulas, as follows.

(D) Let L be a first-order language, let U be a non-empty set, let i/a be an interpretation/assignment from L into U, and let d be the associated designation function. Then the associated valuation function is defined as follows.

Atomic Formulas:

$$\upsilon(\mathbb{P}\tau_1...\tau_n) = i(\mathbb{P})\langle d(\tau_1),...,d(\tau_n)\rangle$$
$$\upsilon(\tau_1=\tau_n) = \mu(=)\langle d(\tau_1),d(\tau_n)\rangle$$

SL-Molecular Formulas:

$$\upsilon(\sim\alpha) = \mu(\sim)\langle\upsilon(\alpha)\rangle$$
$$\upsilon(\alpha\rightarrow\beta) = \mu(\rightarrow)\langle\upsilon(\alpha),\upsilon(\beta)\rangle$$
etc.

Here, the function μ (for 'meaning') assigns a fixed interpretation $\mu(\varepsilon)$ to each special logical symbol; every semantic evaluation υ will assign the same meaning to each special logical symbol.

We have already seen the way '$\sim$', '$\rightarrow$', etc. are evaluated – as truth-functions. In particular,

$\mu(\sim)$ = the truth-function associated with negation
$\mu(\rightarrow)$ = the truth-function associated with the conditional
etc.

Thus far, we have not specifically given the official interpretation of '=', but this is fairly obvious; in particular, we interpret '=' to mean 'is numerically identical to'; alternatively stated, we interpret '=' so that '$\tau_1=\tau_2$' means 'τ_1 and τ_2 are one and the same thing', or simply 'τ_1 <u>is</u> τ_2'.

Formally, $\mu(=)$ is given as follows.

$$\begin{array}{llll} \mu(=)\langle u_1,u_2\rangle & = & T \quad \text{if} & u_1=u_2; \\ & = & F & \text{otherwise.} \end{array}$$

Alternatively, in its set-guise, we have:

$$\mu(=) \quad = \quad \{\langle u,u\rangle : u \in U\}$$

16. Evaluating Quantified Formulas

Our elegant categorial scheme works fine until we come to quantifiers. A universal formula has the form:

$$\forall \nu \mathbb{F}$$

where ν is a variable and $\mathbb{F}$ is a formula. If we treat '$\forall$' as a functor that takes a variable and generates a sentential adverb, then the categorial form looks thus.

$$[\forall\langle\nu\rangle]\langle\mathbb{F}\rangle$$

If we apply the general categorial semantic scheme to this formula we have:

$$[\upsilon(\forall)\langle\upsilon(\nu)\rangle]\langle\upsilon(\mathbb{F})\rangle$$

We have already said that $\upsilon(\mathbb{F})$ and $\upsilon(\forall\nu\mathbb{F})$ are truth-values, this means that $\upsilon(\forall\nu)$ must be a truth-function. But the truth-value of $\forall\nu\mathbb{F}$ is not a function of the truth-value of $\mathbb{F}$. So we are stymied.

We can save the situation by going back and re-doing the entire semantic scheme; this is done in a later chapter. Or, we can say that our semantics is categorial with respect to all syntactic items except quantifiers, which are treated in a non-categorial manner [they are treated as syn-categorimatic].

In this chapter, to keep things as simple as possible, we follow the latter approach. In particular, we treat quantifiers as follows.

$$\begin{array}{lll} \upsilon(\forall\nu\mathbb{F}) & = & \min\{\upsilon'(\mathbb{F}) : \upsilon' \approx_\nu \upsilon\} \\ \upsilon(\exists\nu\mathbb{F}) & = & \max\{\upsilon'(\mathbb{F}) : \upsilon' \approx_\nu \upsilon\} \end{array}$$

Here, $\approx_\nu$ is defined as follows.

$$\upsilon' \approx_\nu \upsilon \quad =_{df} \quad \forall\varepsilon\{\text{atomic}[\varepsilon] \longrightarrow . \ \varepsilon\neq\nu \longrightarrow [\upsilon'(\varepsilon) = \upsilon(\varepsilon)]\}$$

In other words, υ' is "just like" υ in respect to atomic symbols, except insofar as it assigns a different semantic value to variable ν, which of course is the variable bound by $\forall\nu$.

Also, 'min' is short for 'minimum', and 'max' is short for 'maximum'. The implicit ordering, with respect to which these two notions are construed, is obtained by identifying T with 1 and F with 0. In other words, $F\leqslant F<T\leqslant T$.

An alternative, even less algebraic, rendering of the quantifiers goes as follows.

$$\upsilon(\forall \nu \mathbb{F}) = T \quad \text{iff} \quad \forall \upsilon'\{\upsilon' \approx_\nu \upsilon \rightarrow \upsilon'(\mathbb{F})=T\}$$
$$\upsilon(\exists \nu \mathbb{F}) = T \quad \text{iff} \quad \exists \upsilon'\{\upsilon' \approx_\nu \upsilon \ \& \ \upsilon'(\mathbb{F})=T\}$$

The proofs of these two theorems are left as an exercise.

2. Summary of the Quasi-Categorial Semantics for CFOL

1. Semantic Items and their Categories

We begin with the set V of truth-values; i.e., V={T,F}, and a non-empty set U, called the domain, or universe, of discourse. The notion of *semantic item*, or simply *item*, is defined as follows.

every element of V is an item of category V
every element of U is an item of category U;
every k-place function from V into V is an item of category (Vk➔V);
every k-place function from U into U is an item of category (Uk➔U);
every k-place function from U into V is an item of category (Uk➔V);

2. Conventional (Practical) Identifications

k⩾2	Uk➔V	subset of U^k
k=1	U1➔V	subset of U
k=0	U0➔V	element of V

k⩾2	Uk➔U	function from U^k into U
k=1	U1➔U	function from U into U
k=0	U0➔U	element of U

Here, U^k is the set of all k-tuples of elements of U; for example, U^2 is the set of all ordered pairs of elements of U. Also note that $U^0 = \varnothing$ and $U^1 = U$.

$U^2 = \{\langle u_1,u_2\rangle: u_1, u_2 \in U\}$
$U^3 = \{\langle u_1,u_2,u_3\rangle: u_1, u_2, u_3 \in U\}$
etc.

3. Categorial Correspondence

Syntactic Category	Semantic Category
N	U
S	V
Nk➔N	Uk➔U
Nk➔S	Uk➔V
Sk➔S	Vk➔V
quantifiers	not categorial

4. Semantic Evaluations; Algebraic Composition Principle

A semantic evaluation on a quantifier language $\mathbb{L}$ is a function υ that assigns a semantic item of the appropriate category to each well-formed expression of $\mathbb{L}$ (except for quantifiers).

In addition, υ must satisfy the general algebraic-composition principle.

$$\upsilon(\phi\langle\varepsilon_1,\ldots,\varepsilon_k\rangle) \quad = \quad \upsilon(\phi)\langle\upsilon(\varepsilon_1),\ldots,\upsilon(\varepsilon_k)\rangle$$

5. The Sub-Functions of a Semantic Evaluation

Semantic evaluation functions have a number of inter-related sub-functions that work on various special types of grammatical terms.

1. Ordinary Valuation Function – sentences

An ordinary valuation function – also denoted υ – assigns a truth-value to every sentence/formula.

2. Interpretation Function – atomic non-logical (proper) symbols

An interpretation function i assigns an item in the universe to each proper symbol.

$$\text{Proper}[\varepsilon] \longrightarrow \upsilon(\varepsilon) = i(\varepsilon)$$

3. Designation Function – singular terms

A designation function d assigns an element of U to each singular term.

$$\text{SingTerm}[\varepsilon] \longrightarrow \upsilon(\varepsilon) = d(\varepsilon)$$

4. Assignment Function – variables, constants

An assignment function a assigns an element of U to each variable/constant.

$$\text{Variable}[\varepsilon] \vee \text{Constant}[\varepsilon] \;.\longrightarrow \upsilon(\varepsilon) = a(\varepsilon)$$

5. Fixed Logical Meaning Function – special logical symbols

We are interested in those evaluations that respect the meanings of the privileged logical symbols, which in CQL include the SL-operators, the two quantifiers, and the identity sign. Accordingly, these special symbols have a fixed meaning across all semantic evaluations. The function that assigns these fixed meanings is denoted μ.

$$\text{Logical}[\varepsilon] \longrightarrow \upsilon(\varepsilon) = \mu(\varepsilon)$$

6. Determination of Semantic Evaluation

Given the fixed meanings of the special logical symbols, every semantic evaluation is uniquely determined by two sub-functions – the interpretation i, and the assignment function a. In particular,

1. Evaluating Atomic Singular Terms

if τ is atomic, then:

$d(\tau) = i(\tau)$ if τ is a proper noun
$d(\tau) = a(\tau)$ if τ is a variable/constant

2. Evaluating Molecular Singular Terms

$$d(\phi\langle\tau_1,\ldots,\tau_k\rangle) = i(\phi)\langle d(\tau_1),\ldots,d(\tau_k)\rangle$$

3. Evaluating Atomic Formulas

$$\upsilon(\mathbb{P}\langle\tau_1,\ldots,\tau_k\rangle) = i(\mathbb{P})\langle d(\tau_1),\ldots,d(\tau_k)\rangle$$

Given our practical identifications, we have the following in effect.

if $k \geqslant 2$:
$\upsilon(\mathbb{P}\langle\tau_1,\ldots,\tau_k\rangle) = T$ if $\langle d(\tau_1),\ldots,d(\tau_k)\rangle \in i(\mathbb{P})$
$\upsilon(\mathbb{P}\langle\tau_1,\ldots,\tau_k\rangle) = F$ otherwise

if k=1:
$\upsilon(\mathbb{P}\tau) = T$ if $d(\tau) \in i(\mathbb{P})$
$\upsilon(\mathbb{P}\tau) = F$ otherwise

if k=0:
$\upsilon(\mathbb{P}) = i(\mathbb{P})$

Note that identity ('=') is a logical predicate, so its meaning is fixed. In particular,

$$\upsilon(=) = \mu(=) = ID_U$$

Here ID_U is the identity relation on the set U, which is defined in the obvious way.

$$ID_U =_{df} \{\langle u,u\rangle : u \in U\}$$

Alternatively,

$$ID_U =_{df} \{\langle u_1,u_2\rangle : u_1,u_2 \in U \ \& \ u_1=u_2\}$$

The characteristic function – also denoted ID_U – is defined as follows.

$$\begin{array}{lllll} ID_U\langle u_1,u_2\rangle & = & T & \text{if} & u_1=u_2 \\ & = & F & & \text{otherwise} \end{array}$$

4. Evaluating SL Compounds

$$\begin{array}{ll} \upsilon(\sim\mathbb{F}) & = \mu(\sim)\langle\upsilon(\mathbb{F})\rangle \\ \upsilon(\mathbb{F}\rightarrow\mathbb{G}) & = \mu(\rightarrow)\langle\upsilon(\mathbb{F}),\upsilon(\mathbb{G})\rangle \\ \upsilon(\mathbb{F}\&\mathbb{G}) & = \mu(\&)\langle\upsilon(\mathbb{F}),\upsilon(\mathbb{G})\rangle \\ \upsilon(\mathbb{F}\vee\mathbb{G}) & = \mu(\vee)\langle\upsilon(\mathbb{F}),\upsilon(\mathbb{G})\rangle \\ \upsilon(\mathbb{F}\leftrightarrow\mathbb{G}) & = \mu(\leftrightarrow)\langle\upsilon(\mathbb{F}),\upsilon(\mathbb{G})\rangle \end{array}$$

Here, if ε is an SL-operator, then $\mu(\varepsilon)$ is its fixed meaning, which is a truth-function. If we use the same symbol for both the connective and its truth function, and we write two-place functions/functors in infix notation, we can rewrite this in the familiar algebraic form.

$$\begin{array}{ll} \upsilon(\sim\mathbb{F}) & = \sim\upsilon(\mathbb{F}) \\ \upsilon(\mathbb{F}\rightarrow\mathbb{G}) & = \upsilon(\mathbb{F})\rightarrow\upsilon(\mathbb{G}) \\ \upsilon(\mathbb{F}\&\mathbb{G}) & = \upsilon(\mathbb{F})\&\upsilon(\mathbb{G}) \\ \upsilon(\mathbb{F}\vee\mathbb{G}) & = \upsilon(\mathbb{F})\vee\upsilon(\mathbb{G}) \\ \upsilon(\mathbb{F}\leftrightarrow\mathbb{G}) & = \upsilon(\mathbb{F})\leftrightarrow\upsilon(\mathbb{G}) \end{array}$$

5. Evaluating Quantified Formulas

$$\begin{array}{ll} \upsilon(\forall\nu\mathbb{F}) & = \min\{\upsilon'(\mathbb{F}) : \upsilon' \approx_\nu \upsilon\} \\ \upsilon(\exists\nu\mathbb{F}) & = \max\{\upsilon'(\mathbb{F}) : \upsilon' \approx_\nu \upsilon\} \end{array}$$

Here, $\approx_\nu$ is defined as follows.

$$\upsilon' \approx_\nu \upsilon \quad =_{df} \quad \forall\varepsilon\{\text{atomic}[\varepsilon] \longrightarrow . \ \varepsilon\neq\nu \longrightarrow [\upsilon'(\varepsilon) = \upsilon(\varepsilon)]\}$$

In other words, υ' is "just like" υ in respect to atomic symbols, except insofar as it assigns a different semantic value to variable ν, which of course is the variable bound by $\forall\nu$.

Also, 'min' is short for 'minimum', and 'max' is short for 'maximum'. The implicit ordering, with respect to which these two notions are construed, is obtained by identifying T with 1 and F with 0. In other words, $F\leqslant F<T\leqslant T$.

An alternative, even less algebraic, rendering of the quantifiers goes as follows.

$$\begin{array}{llll} \upsilon(\forall\nu\mathbb{F}) & = T & \text{iff} & \forall\upsilon'\{\upsilon' \approx_\nu \upsilon \longrightarrow \upsilon'(\mathbb{F})=T\} \\ \upsilon(\exists\nu\mathbb{F}) & = T & \text{iff} & \exists\upsilon'\{\upsilon' \approx_\nu \upsilon \ \& \ \upsilon'(\mathbb{F})=T\} \end{array}$$

3. Appendix – Models and Satisfaction

1. Introduction

In the parent chapter, we have examined a mostly-categorial semantics for first-order logic, and in the first appendix, we have examined a fully-categorial semantics for first-order logic. Although the categorial approach has many advantages in terms of mathematical elegance and rigor, it is not the way logicians have traditionally done semantics. For this reason, in the current appendix, we briefly look at the completely traditional (non-categorial) approach to semantics.

2. Models and Interpretations in Sentential Logic

We start by going back and redoing the semantics of SL in the non-categorial manner. First, we define the notion of *interpretation*.

> Def
>
> Let $\mathbb{L}$ be the language of CSL. Then an *interpretation* on/of $\mathbb{L}$ is, by definition, any function I that assign a truth-value to every atomic formulas of $\mathbb{L}$.

Notice that this is consistent with our usage in the parent chapter, where we define an interpretation as a function that assigns semantic values to all (and only) atomic non-logical symbols. In SL, these are precisely the atomic formulas.

Intimately associated with the notion of interpretation is the notion of *model*, which is defined as follows.

> Def
>
> Let $\mathbb{L}$ be the language of CSL. Then a *model* on/of $\mathbb{L}$ is, by definition, any infinite sequence $\langle v_1, v_2, \ldots\rangle$ of truth-values.

A model is semantically useful precisely because we presume a fixed enumeration $\langle A_1, A_2, \ldots\rangle$ of the atomic formulas of $\mathbb{L}$. In particular, the i-th truth-value v_i is understood to be the truth-value of the i-th atomic formula.

If we presuppose a given fixed enumeration $\langle A_1, A_2, \ldots\rangle$ of atomic formulas, there is no practical difference between a model and an interpretation. Given an interpretation I, one can easily construct the associated model $\mathcal{M}$(I) as follows.

$$\mathcal{M}(\mathrm{I}) \;=\; \langle \mathrm{I}(A_i), \mathrm{I}(A_2), \ldots\rangle$$

Similarly, given any model $\mathcal{M}$ [$= \langle v_1, v_2, \ldots\rangle$], one can easily construct the associated interpretation $\mathrm{I}_{\mathcal{M}}$ as follows.

$$\mathrm{I}_{\mathcal{M}}(A_i) \;=\; v_i \qquad i = 1, 2, \ldots$$

3. Truth in an Interpretation/Model in SL

Given an interpretation I, we can define the notion of *truth in an interpretation* as follows.

Def

Let $\mathbb{L}$ be the language of CSL. Let I be an interpretation of $\mathbb{L}$. Define the predicate '$I \models \phi$' (read "I satisfies ϕ", or "I verifies ϕ", or "ϕ is true in I") inductively as follows.

if ϕ is atomic, then

$I \models \phi$ iff $I(\phi) = T$

if ϕ is molecular, then there are five cases:

$I \models \sim\alpha$ iff $I \not\models \alpha$

$I \models \alpha \& \beta$ iff $I \models \alpha$ and $I \models \beta$

$I \models \alpha \vee \beta$ iff $I \models \alpha$ or $I \models \beta$

$I \models \alpha \rightarrow \beta$ iff $I \not\models \alpha$ or $I \models \beta$

$I \models \alpha \leftrightarrow \beta$ iff $I \models \alpha$ and $I \models \beta$ .or. $I \not\models \alpha$ and $I \not\models \beta$

We can of course give a corresponding definition of *truth in a model* as follows.

Def

Let $\mathbb{L}$ be the language of CSL. Let $\mathcal{M}$ [$= \langle v_1, v_2, \ldots \rangle$] be a model of $\mathbb{L}$. Define the predicate '$\mathcal{M} \models \phi$' (read "$\mathcal{M}$ satisfies ϕ", or "$\mathcal{M}$ verifies ϕ", or "ϕ is true in $\mathcal{M}$") inductively as follows.

if ϕ is atomic,
then $\phi = A_i$ (for exactly one i),
and

$\mathcal{M} \models \phi$ iff $v_i = T$

if ϕ is molecular, then there are five cases:

$\mathcal{M} \models \sim\alpha$ iff $\mathcal{M} \not\models \alpha$

$\mathcal{M} \models \alpha\&\beta$ iff $\mathcal{M} \models \alpha$ and $\mathcal{M} \models \beta$

$\mathcal{M} \models \alpha \vee \beta$ iff $\mathcal{M} \models \alpha$ or $\mathcal{M} \models \beta$

$\mathcal{M} \models \alpha \rightarrow \beta$ iff $\mathcal{M} \not\models \alpha$ or $\mathcal{M} \models \beta$

$\mathcal{M} \models \alpha \leftrightarrow \beta$ iff $\mathcal{M} \models \alpha$ and $\mathcal{M} \models \beta$ .or. $\mathcal{M} \not\models \alpha$ and $\mathcal{M} \not\models \beta$

4. Validity in the Model Framework

Recall that we defined validity relative to a class $\mathbb{V}$ of admissible valuations. We can also define validity relative to a class $\mathbb{I}$ of all interpretations of $\mathbb{L}$, or relative to the class $\mathbb{M}$ of all models of $\mathbb{L}$.

Def

Let $\mathbb{L}$ be the language of CSL. Let $\mathbb{I}$ be the set of all interpretations on $\mathbb{L}$. Define the predicates '$\models \alpha$' and '$\Gamma \models \alpha$' as follows.

$\models \alpha$ iff for every interpretation I, $I \models \alpha$

$\Gamma \models \alpha$ iff for every interpretation I,
if $I \models \gamma$ for every $\gamma \in \Gamma$,
then $I \models \alpha$

Def

Let $\mathbb{L}$ be the language of CSL. Let $\mathbb{M}$ be the set of all models on $\mathbb{L}$. Define the predicates '$\models\alpha$' and '$\Gamma\models\alpha$' as follows.

$\models\alpha$ iff for every model $\mathcal{M}$, $\mathcal{M}\models\alpha$

$\Gamma\models\alpha$ iff for every model $\mathcal{M}$,
if $\mathcal{M}\models\gamma$ for every $\gamma\in\Gamma$,
then $\mathcal{M}\models\alpha$

5. Models (and Interpretations) in the Context of First-Order Logic

Next, we turn to models of first-order languages. Notice that, whereas in SL, the notion of model and interpretation are interchangeable, in FOL it is customary to use the notion of *model*, which will conceptually include the notion of *interpretation*.

Def

Let $\mathbb{L}$ be a first-order language. Then a *model* of $\mathbb{L}$ is a structure $\langle U, I\rangle$, where U is a non-empty set – called the *domain* or *universe* of discourse – and I is an *interpretation function*. Here, an interpretation function is a function that assigns a categorially-appropriate set-theoretic object "over" U to each proper (i.e., non-logical atomic) symbol of $\mathbb{L}$.

Recall that the "proper" symbols of a FOL are those symbols that uniquely identify the language; the remaining symbols are common to all FOL's. The term 'categorially-appropriate' is defined (in a somewhat ad hoc manner) as follows.

Def

Let $\mathbb{L}$ be a FOL, and let I be a function from the set of proper symbols of $\mathbb{L}$. Then I is an *interpretation* iff the following are true.

if ε is a proper noun, then $I(\varepsilon) \in U$
if ε is a 1-place function sign, then $I(\varepsilon)$ is a function from U into U
if ε is a k-place function sign, then $I(\varepsilon)$ is a function from U^k into U
if ε is a 1-place predicate, then $I(\varepsilon)$ is a subset of U
if ε is a k-place predicate, then $I(\varepsilon)$ is a subset of U^k

Thus, the notion of interpretation here is very similar to the notion of interpretation in the parent chapter. In particular, an interpretation assigns a set-theoretic object to every proper symbol of $\mathbb{L}$.

6. Assignment Functions; Designation Functions

As in the case of valuation semantics, we employ assignment functions to evaluate expressions containing variables and constants. As before, an assignment function assigns to every variable and constant an element of the domain U.

As in the case of valuation semantics, every combination of interpretation/assignment gives rise to an associated designation function, which assigns a denotation to every singular term in $\mathbb{L}$. It is defined as follows.

If τ is a proper noun, then

$$d(\tau) = \mathrm{I}(\tau)$$

If τ is a variable or constant, then

$$d(\tau) = a(\tau)$$

If τ is molecular, then it has the form $\phi\langle\tau_1, \ldots, \tau_k\rangle$, and

$$d(\tau) = \mathrm{I}(\phi)\langle d(\tau_1), \ldots, d(\tau_k)\rangle$$

Given a model $\mathcal{M}$, we can define the notion of $\mathcal{M}$-admissible designation function as follows.

Def

Let $\mathbb{L}$ be a FOL, and let $\mathcal{M} = \langle \mathrm{U}, \mathrm{I}\rangle$ be a model of $\mathbb{L}$. Let d be a function from the singular terms of $\mathbb{L}$ into U. Then d is said to be *$\mathcal{M}$-admissible* iff there is an assignment function a from $\mathbb{L}$ into U such that d is the designation function determined by I/a.

7. Satisfaction by a Designation Function

Once we have the denotations of all the singular terms of $\mathbb{L}$, we can discuss the notion of *satisfaction*. In particular, satisfaction is defined inductively as follows.

Def

Let $\mathbb{L}$ be a FOL, let $\mathcal{M}$ be a model of $\mathbb{L}$, and let d be an $\mathcal{M}$-admissible designation function. Define '$d \vDash \phi$' as inductively as follows.

If ϕ is atomic, then $\phi = P\langle\tau_1, \ldots, \tau_k\rangle$, in which case

$$d \vDash \phi \quad \text{iff} \quad \langle d(\tau_1), \ldots, d(\tau_k)\rangle \in I(P)$$

or $\phi = [\tau_1 = \tau_2]$, in which case

$$d \vDash \phi \quad \text{iff} \quad d(\tau_1) = d(\tau_k).$$

If ϕ is an SL-molecule, then there are five cases to consider:

$$\begin{array}{lll} d \vDash \sim\alpha & \text{iff} & d \nvDash \alpha \\ d \vDash \alpha\&\beta & \text{iff} & d \vDash \alpha \text{ and } d \vDash \beta \\ d \vDash \alpha\vee\beta & \text{iff} & d \vDash \alpha \text{ or } d \vDash \beta \\ d \vDash \alpha\rightarrow\beta & \text{iff} & d \nvDash \alpha \text{ or } d \vDash \beta \\ d \vDash \alpha\leftrightarrow\beta & \text{iff} & d \vDash \alpha \text{ and } d \vDash \beta, \text{ or } d \nvDash \alpha \text{ and } d \nvDash \beta \end{array}$$

If ϕ is a quantified formula, then there are two cases to consider:

$$\begin{array}{lll} d \vDash \forall\nu\psi & \text{iff} & d' \vDash \psi, \text{ for every } d' \approx_\nu d \\ d \vDash \exists\nu\psi & \text{iff} & d' \vDash \psi, \text{ for some } d' \approx_\nu d \end{array}$$

Here, the $\approx$ relation is defined pretty much as before.

$$d_1 \approx_\nu d_2 \quad \text{iff} \quad \forall\varepsilon\{\text{Atomic}[\varepsilon] \;\&\; \varepsilon \neq \nu \;.\!\rightarrow\; d_1(\varepsilon) = d_2(\varepsilon)$$

8. Satisfaction by a Model

Satisfaction by a model is defined in a natural manner based on the notion of satisfaction by a designation function.

Def

Let $\mathbb{L}$ be a FOL, and let $\mathcal{M}$ be a model of $\mathbb{L}$. Then

$$\mathcal{M} \vDash \phi \quad \text{iff} \quad d \vDash \phi \quad \text{for every } \mathcal{M}\text{-admissible } d$$