13 The Syntax of Classical First-Order Logic

1. Introduction

So far, we have examined classical sentential logic (CSL), both from the semantic (model theoretic) viewpoint, and from the axiomatic (proof theoretic) viewpoint. We now turn to classical first-order logic (CFOL). In the current chapter, we describe the syntax of CFOL.

In an earlier chapter, on formal languages, we described full first-order languages, which include quantification, identity, and definite descriptions. Since descriptions present many semantic difficulties, we are going to concentrate on first-order logic without descriptions.

2. First-Order Languages – The Category

First-order languages are a kind of formal language. In this section, we describe the general category. Later, we examine a specific example.

1. Vocabulary

1. Logical Vocabulary

variables; a denumerable list
constants; a denumerable list (optional)
special logical symbols:
 quantifiers: ∀, ∃
 connectives: ∼, →, ↔, &, ∨
 identity sign: =
parentheses: (,)

The vocabulary of a FOL consists of an expressively complete set of proper logical symbols [e.g., ∀, ∼, →, =].

2. Non-Logical Vocabulary (Proper Symbols)

proper nouns*
function signs; 0-place, 1-place, 2-place, etc.
predicates; 0-place, 1-place, 2-place, etc.

The vocabulary of a FOL consists of zero or more of each of these categories. These are the proper symbols of a given FOL. Whereas every FOL has the same logical symbols, they differ in regard to proper symbols.

*Proper nouns are redundant; if we wish, we may simply regard proper nouns as 0-place function signs. Nevertheless, we include them for descriptive completeness.

2. Rules of Formation

1. Singular Terms

every variable/constant is a singular term;
every proper noun is a singular term;
if ϕ is an n-place function sign, and $\tau_1,\ldots,\tau_n$ are singular terms, then $\phi\langle\tau_1,\ldots,\tau_n\rangle$ is a singular term;
nothing else is a singular term.

2. Atomic Formulas

if $\mathbb{P}$ is an n-place predicate and $\tau_1,\ldots,\tau_n$ are singular terms, then $\mathbb{P}\langle\tau_1,\ldots,\tau_n\rangle$ is a formula;
nothing else is an atomic formula.

3. Formulas

if $\mathbb{F}$ and $\mathbb{G}$ are formulas, then so are $\&\langle\mathbb{F},\mathbb{G}\rangle$, $\vee\langle\mathbb{F},\mathbb{G}\rangle$, $\rightarrow\langle\mathbb{F},\mathbb{G}\rangle$, $\leftrightarrow\langle\mathbb{F},\mathbb{G}\rangle$, $\sim\langle\mathbb{F}\rangle$;
if $\mathbb{F}$ is a formula, and ν is a variable, then $\forall\langle\nu,\mathbb{F}\rangle$ and $\exists\langle\nu,\mathbb{F}\rangle$ are formulas;
nothing else is a formula.

Note carefully: In general, where $\acute{\omega}$ is a functor, and ε_1, …, ε_k are argument expressions of the appropriate category, the metalinguistic expression '$\acute{\omega}\langle\varepsilon_1,\ldots,\varepsilon_n\rangle$' refers to the expression obtained when the functor $\acute{\omega}$ is applied to those arguments, without regard to how functor application is in fact syntactically implemented by the particular language. The actual syntactic implementation may be prefix, infix, postfix, or some other scheme.

3. The Official Formal Language of Classical First-Order Logic

The previous section offers a very general description of first-order languages. In the current section, we offer a concrete example of a first-order language – the language of classical first-order logic (CFOL). The example is concrete in the sense that the actual syntactic elements are orthographically identified.

In particular, the official language of CFOL is a formal language with the following rules of formation. We use quote-plus notation.

1. Vocabulary

1. Logical Vocabulary

Variables:

'x' is a variable;
if σ is a variable, then so is σ+'♯';
nothing else is a variable.

Constants:

'a' is a constant;

if σ is a constant, then so is σ+‘♯’;
nothing else is a constant.

quantifier: ‘∀’
connectives: ‘∼’, ‘→’
parentheses: ‘(’, ‘)’

2. Non-Logical Vocabulary

Proper Nouns:

‘n’ is a proper noun
if σ is a proper noun, then so is σ+‘♯’;
nothing else is a proper noun.

Function Signs:

0-place:

‘f’ is a 0-place function sign;
if σ is a 0-place function sign, then so is σ+‘♯’;
nothing else is a 0-place function sign.

1-place:

‘f♭’ is a 1-place function sign;
if σ is a 1-place function sign, then so is σ+‘♯’;
nothing else is a 1-place function sign.

2-place:

‘f♭♭’ is a 2-place function sign;
if σ is a 2-place function sign, then so is σ+‘♯’;
nothing else is a 2-place function sign.

3-place, 4-place, etc., function signs.

Predicates:

0-place:

‘P’ is a 0-place predicate;
if σ is a 0-place predicate, then so is σ+‘♯’;
nothing else is a 0-place predicate.

1-place:

‘P♭’ is a 1-place predicate;
if σ is a 1-place predicate, then so is σ+‘♯’;
nothing else is a 1-place predicate.

2-place:

‘P♭♭’ is a 2-place predicate;
if σ is a 2-place predicate, then so is σ+‘♯’;
nothing else is a 2-place predicate.

3-place, 4-place, etc., predicates.

2. Rules of Formation

1. Singular Terms

every variable is a singular term;

every constant is a singular term;

every proper noun is a singular term;

if φ is an n-place function sign, and $\tau_1,...,\tau_n$ are singular terms, then $\phi+\tau_1+...+\tau_n$ is a singular term;

nothing else is a singular term.

2. Atomic Formulas

if τ_1 and τ_2 are singular terms then ‘(’ + τ_1 + ‘=’ + τ_2 + ‘)’ is an atomic formula

if $\mathbb{P}$ is an n-place non-logical predicate and $\tau_1,...,\tau_n$ are singular terms, then $\mathbb{P}+\tau_1+...+\tau_n$ is an atomic formula;

nothing else is an atomic formula.

3. Formulas

every atomic formula is a formula;

if $\mathbb{F}$ and $\mathbb{G}$ are formulas, then so are ‘(’+$\mathbb{F}$+‘→’+$\mathbb{G}$+‘)’ and ‘∼’+$\mathbb{F}$;

if $\mathbb{F}$ is a formula, and ν is a variable, then ‘∀’+ν+$\mathbb{F}$ is a formula;

nothing else is a formula.

4. Definitions of Non-Primitive Logical Signs

$\exists\nu\alpha$	$=_{df}$	$\sim\forall\nu\sim\alpha$
$(\alpha\&\beta)$	$=_{df}$	$\sim(\alpha\rightarrow\sim\beta)$
$(\alpha\vee\beta)$	$=_{df}$	$(\sim\alpha\rightarrow\beta)$
$(\alpha\leftrightarrow\beta)$	$=_{df}$	$(\alpha\rightarrow\beta)\&(\beta\rightarrow\alpha)$

5. Metalinguistic Notational Shorthand

x_0	$=_{df}$	‘x’
x_1	$=_{df}$	‘x♯’
x_2	$=_{df}$	‘x♯♯’
etc.		

f^0_0	$=_{df}$	‘f’
f^0_1	$=_{df}$	‘f♯’
f^0_2	$=_{df}$	‘f♯♯’
etc.		

f^1_0	$=_{df}$	‘f♭’
f^1_1	$=_{df}$	‘f♭♯’
f^1_2	$=_{df}$	‘f♭♯♯’
etc.		

f^2_0	$=_{df}$	‘f♭♭’
f^1_1	$=_{df}$	‘f♭♭♯’
f^1_2	$=_{df}$	‘f♭♭♯♯’
etc.		

c_0	$=_{df}$	‘a’
c_1	$=_{df}$	‘a♯’
c_2	$=_{df}$	‘a♯♯’
etc.		

P^0_0	$=_{df}$	‘P’
P^0_1	$=_{df}$	‘P♯’
P^0_2	$=_{df}$	‘P♯♯’
etc.		

P^1_0	$=_{df}$	‘P♭’
P^1_1	$=_{df}$	‘P♭♯’
P^1_2	$=_{df}$	‘P♭♯♯’
etc.		

P^2_0	$=_{df}$	‘P♭♭’
P^2_1	$=_{df}$	‘P♭♭♯’
P^2_2	$=_{df}$	‘P♭♭♯♯’
etc.		

6. Further Natural Notational Shorthand

In order to write formulas in a manner similar to elementary logic, we also adopt the following shorthand conventions.

‘b’	$=_{df}$	‘a_2’
‘c’	$=_{df}$	‘a_3’
“etc.”		
‘y’	$=_{df}$	‘x_2’
‘z’	$=_{df}$	‘x_3’
“etc.”		
‘f’	$=_{df}$	‘f^1_1’
‘g’	$=_{df}$	‘f^1_2’
“etc.”		
‘F’	$=_{df}$	‘P^1_1’
‘G’	$=_{df}$	‘P^1_2’
“etc.”		
‘R’	$=_{df}$	‘P^2_1’
‘S’	$=_{df}$	‘P^2_2’
“etc.”		

7. Further Syntactic Issues Pertaining to CFOL

1. Symbols versus Occurrences of Symbols.

How many words are there in the following brief passage?

the small dog barked at the large dog

Well, it depends on what you mean. This question is ambiguous between the following two different questions. (1) How many different (unique) words are used in this passage? (2) How long is this passage, measured in words, or how many *word occurrences* are there in this passage? The answer to the first question is 6; the answer to the second question is 8. For example, the word ‘the’ appears 2 times; which is to say that there are 2 *occurrences* of the word ‘the’ in this passage.

Just as a given word of English (e.g., ‘the’) can occur many times in a given sentence (or paragraph) of English, a given logic symbol can occur many times in a given formula. And in particular, a given variable can occur many times in a formula. Consider the following examples of occurrences of variables.

(1)	Fx	‘x’ occurs once	[or: there is one occurrence of ‘x’;]
(2)	Rxy	‘x’ occurs once;	‘y’ occurs once;
(3)	$Fx \rightarrow Hx$	‘x’ occurs twice;	
(4)	$\forall x(Fx \rightarrow Hx)$	‘x’ occurs three times.	

We also speak the same way about occurrences of other symbols and combinations of symbols. So, for example, we can speak of occurrences of ‘$\sim$’, or occurrences of ‘$\forall x$’.

Mathematically, if we identify expressions as strings of symbols, then we can identify an occurrence of an expression ε as an ordered pair ε/k, where k is the ordinal address of the occurrence of ε relative to all occurrences of ε within the overall string in question. In other words, we can refer to the first occurrence of ‘x’ as ‘x’/1, the second occurrence of ‘x’ as ‘x’/2, etc.

2. Quantifier Scope.

Df

The ***scope*** of an occurrence of a quantifier in an expression ε is, by definition, the smallest sub-formula of ε that contains that occurrence.

Examples:

(1)	$\forall xFx \rightarrow Fa$	the scope of the only occurrence of ‘∀x’ is:	‘$\forall xFx$’
(2)	$\forall x(Fx \rightarrow Gx)$	the scope of the only occurrence of ‘∀x’ is:	‘$\forall x(Fx \rightarrow Gx)$’
(3)	$Fa \rightarrow \forall x(Gx\rightarrow Hx)$	the scope of the only occurrence of ‘∀x’ is:	‘$\forall x(Gx \rightarrow Hx)$’

As a somewhat more complicated example, consider the following.

(4) $\forall x(\forall yRxy \rightarrow \forall zRzx)$

the scope of the only occurrence of ‘∀x’ is ‘$\forall x(\forall yRxy \rightarrow \forall zRzx)$’
the scope of the only occurrence of ‘∀y’ is ‘$\forall yRxy$’
the scope of the only occurrence of ‘∀z’ is ‘$\forall zRzx$’

As a still more complicated example, consider the following.

(5) $\forall x[\forall xFx \rightarrow \forall y(\forall yGy \rightarrow \forall zRxyz)]$;

the scope of the first occurrence of ‘∀x’ is the whole formula;
the scope of the second occurrence of ‘∀x’ is ‘$\forall xFx$’;
the scope of the first ∀y occurrence of is ‘$\forall y(\forall yGy \rightarrow \forall zRxyz)$’;
the scope of the second occurrence of ‘∀y’ is ‘$\forall yGy$’;
the scope of the only occurrence of ‘∀z’ is ‘$\forall zRxyz$’.

3. Government and Binding

Df

for any variable ν, ∀ν governs the variable ν
for any variable ν, ∀ν governs the variable ν

Df

An occurrence o_1 of a quantifier *binds* an occurrence o_2 of a variable if and only if
(1) the quantifier governs the variable, and
(2) o_2 is contained within the scope of o_1.

Df

An occurrence o_1 of a quantifier Q *truly binds* an occurrence o_2 of a variable if and only if
(1) o_1 binds o_2, and
(2) o_1 is inside the scope of every occurrence of Q that binds o_2.

Example

$\forall x(Fx \rightarrow \forall xGx)$;

In this formula the first '$\forall$x' binds every occurrence of 'x', but it only truly binds the first two occurrences; on the other hand, the second '$\forall$x' truly binds the last two occurrences of 'x'.

4. Free versus Bound Occurrences of Variables

Every given occurrence of a given variable is either *free* or *bound* in an expression ε in which it occurs.

Df

An occurrence of a variable in a formula $\mathbb{F}$ is *bound* in $\mathbb{F}$ if and only if that occurrence is bound by some quantifier occurrence in $\mathbb{F}$.

Df

An occurrence of a variable in a formula $\mathbb{F}$ is *free* in $\mathbb{F}$ if and only if that occurrence is not bound in $\mathbb{F}$.

Examples

(1) Fx:

the one and only occurrence of 'x' is free in this formula;

(2) $\forall x(Fx \rightarrow Gx)$:

all three occurrences of 'x' are bound by '$\forall$x';

(3) $Fx \rightarrow \forall xGx$:

the first occurrence of ‘x’ is free; the remaining two occurrences are bound.

(4) $\forall x(Fx \rightarrow \forall xGx)$:

the first two occurrences of ‘x’ are bound by the first ‘∀x’; the second two are bound by the second ‘∀x’.

(5) $\forall x(\forall yRxy \rightarrow \forall zRzx)$:

every occurrence of every variable is bound.

Notice in example (4) that the variable ‘x’ occurs within the scope of two different occurrences of ‘∀x’. It is only the innermost occurrence of ‘∀x’ that truly binds the last two occurrences of the variable, however. The other occurrence of ‘∀x’ truly binds the first occurrence of ‘x’ but none of the remaining ones.

5. Closed Formulas versus Open Formulas

Df

A formula $\mathbb{F}$ is *closed* if and only if no variable occurs free in $\mathbb{F}$.
A singular term τ is *closed* if and only if no variable occurs free in τ.

A formula/term is *open* if and only if it is not closed.

6. Substitution Instances; Free For

Df

Let $\mathbb{F}$ be any formula, let v be any variable, and let τ be any singular term. Then:

$\mathbb{F}[\tau/v]$ $=_{df}$ the result of substituting τ for every occurrence of v that is free in $\mathbb{F}$

v is said to be *free for* τ *in* $\mathbb{F}$ if and only if every occurrence of every variable that is free in τ is free in $\mathbb{F}[\tau/v]$ [Note: if we do not have definite descriptions, then every variable occurs free in every singular term.]

if v is free for τ in $\mathbb{F}$, then
$\mathbb{F}[\tau/v]$ is said to be a *substitution instance* of $\mathbb{F}$.

Examples:

if τ is a closed singular term, then no variable is free in τ, so every variable is free for τ in $\mathbb{F}$

if no variable in τ occurs in $\mathbb{F}$, no variable in τ is bound in $\mathbb{F}[\tau/v]$, so v is free for τ in $\mathbb{F}$

'x' is not free for 'y' in '∃yRxy', since

∃yRxy[y/x] = ∃yRyy,
and 'y' occurs free in 'y', but does not occur free in '∃yRyy'

7. Alphabetic Variants

Def [inductive]

Let $\mathbb{F}$ be a closed formula. Let π be any permutation of the set of variables. Let F* be the result of substituting $\pi(v)$ for each occurrence of each variable v that occurs in $\mathbb{F}$. Then F* is an *alphabetic variant* of $\mathbb{F}$.

If $\mathbb{F}$ is a subformula of $\mathbb{F}'$, and F* is an alphabetic variant of $\mathbb{F}$, then $\mathbb{F}'[\mathbb{F}^*/\mathbb{F}]$ is an alphabetic variant of $\mathbb{F}'$

Nothing else is an alphabetic variant.

Examples:

(1)	∀xFx; ∀yFy; ∀zFz	are alphabetic variants of each other;
(2)	∀xRxy; ∀zRzy; ∀wRwy	are alphabetic variants of each other;
(3)	∀xFx → ∀xHx; ∀yFy → ∀zHz	are alphabetic variants of each other;
(4)	∀x(Fx → ∀xHx); ∀x(Fx → ∀yHy)	are alphabetic variants of each other.
(5)	Fx; Fy; Fz	are <u>not</u> alphabetic variants of each other.