Chapter 7

The Language

In this and subsequent chapters, our metalanguage makes use of informal notions and principles about numbers and sets so as to more precisely articulate and render certain definitions. These are *not* primitive notions or principles of our metaphysical system; ultimately, they are to be understood by using object theory to analyze the language of mathematics, as we shall do in several later chapters.

7.1 Metatheoretical Definitions

We begin with the definitions that create a particular second-order language in all of its details.

(**1**) **Definitions:** Simple Terms. A *simple term* of our language is any expression that is a *simple individual term* or a *simple n-place relation term* ($n \geq 0$), where these are listed as follows:

(.1) Simple Individual Terms: (Less Formal)
Individual Constants (Names):
$a_1, a_2, \ldots$ (a, b, c, d, e)
Individual Variables:
$x_1, x_2, \ldots$ (x, y, z, u, v, w)

(.2) Simple n-place Relation Terms ($n \geq 0$):
n-place Relation Constants (Names)
$P^n_1, P^n_2, \ldots$ $(P^n, Q^n, R^n, S^n, T^n)$
n-place Relation Variables:
$F^n_1, F^n_2, \ldots$ $(F^n, G^n, H^n, I^n, J^n)$

(.3) Distinguished 1-place Relation Constant:
$E!$ (read: 'being concrete' or 'concreteness')

A *constant* is any expression that is either an individual constant or an n-place relation constant ($n \geq 0$). A *variable* is any expression that is either an individual variable or an n-place relation variable ($n \geq 0$). The expressions listed in the column labeled 'Less Formal' are often used as replacements to facilitate readability. It is assumed that we never run out of primitive constants and variables, despite the fact that the less formal alternatives appear in a finite list.

(2) **Definitions:** Other Primitive Expressions of the Language. The simple terms of our language are primitive expressions. The other primitive expressions of our language are listed here. In what follows, we use α, β, γ as metavariables ranging over all variables and use μ, ν (Greek *mu*, *nu*), sometimes with a prime or an index, as metavariables ranging just over primitive individual variables. Thus, our language includes the following additional primitive expressions:

(.1) Unary Formula-Forming Operators:
$\neg$ ('it is not the case that' or 'it is false that')
$\mathcal{A}$ ('actually' or 'it is actually the case that')
$\Box$ ('necessarily' or 'it is necessary that')

(.2) Binary Formula-Forming Operator:
$\rightarrow$ ('if . . . , then . . . ')

(.3) Variable-Binding Formula-Forming Operator:
$\forall\alpha$ ('every α is such that')
for every variable α

(.4) Variable-Binding Individual-Term-Forming Operator:
$\iota\nu$ ('the ν such that')
for every individual variable ν

(.5) Variable-Binding n-place Relation-Term-Forming Operators ($n \geq 0$):
$\lambda\nu_1 \ldots \nu_n$ ('being individuals $\nu_1 \ldots \nu_n$ such that')
for any distinct individual variables $\nu_1, \ldots, \nu_n$
When $n = 0$, the λ binds no variables and is read as 'that'

These primitive, syncategorematic expressions are referenced in the definition of the syntax of our language and become parts of the complex formulas and complex terms defined there.[78] In what follows, we sometimes call $\neg$ the *nega-*

[78] A syncategorematic expression is an expression that is neither a term, i.e., an expression that is assigned a denotation, nor a formula, i.e., an expression that is assigned truth conditions. Thus these are expressions that aren't assigned a semantic value in and of themselves.

tion operator, $\mathcal{A}$ the *actuality* operator, $\Box$ the *necessity* operator, $\rightarrow$ the *conditional*; $\forall\alpha$ a *universal quantifier*, and $\iota\nu$ a *definite description* operator. There is no special name for $\lambda\nu_1 \ldots \nu_n$ operators.

(**3**) **Definitions:** Syntax of the Language. We present the syntax of our language by a simultaneous recursive definition of the following four kinds of expressions: *individual term*, *n-place relation term*, *formula*, and *propositional formula*. Before we give the definition, note that we shall be defining two kinds of (complex) formulas and we can intuitively describe the two kinds as ones that have encoding subformulas and ones that don't.[79] The latter are designated as *propositional formulas*. In what follows, we use the parenthetical expression '(propositional)' strategically so that we don't have to repeat clauses in the definition of *formula* to produce clauses in the definition of *propositional formula*. This technique is explained in more detail in Remark (4), which follows the definition.

(.1) Every simple individual term is an individual term and every simple n-place relation term is an n-place relation term ($n \geq 0$).

(.2) If Π^n is any n-place relation term and $\kappa_1, \ldots, \kappa_n$ are any individual terms, then where $n \geq 1$,

(.a) $\Pi^n\kappa_1 \ldots \kappa_n$ is a (propositional) formula ('$\kappa_1, \ldots, \kappa_n$ exemplify Π^n')

(.b) $\kappa_1\Pi^1$ is a formula ('κ_1 encodes Π^1')

and where Π^0 is any 0-place relation term,

(.c) Π^0 is a (propositional) formula ('Π^0 is true')

(.3) If φ, ψ are (propositional) formulas and α any variable, then $(\neg\varphi)$, $(\varphi \rightarrow \psi)$, $\forall\alpha\varphi$, $(\Box\varphi)$, and $(\mathcal{A}\varphi)$ are (propositional) formulas.

(.4) If φ is any formula and ν any individual variable, then $\iota\nu\varphi$ is an individual term

(.5) If φ is any propositional formula and $\nu_1, \ldots, \nu_n$ are any distinct individual variables ($n \geq 0$), then

(.a) $[\lambda\nu_1 \ldots \nu_n \varphi]$ is an n-place relation term, and

(.b) φ itself is a 0-place relation term[80]

[79] This description is intuitive because the notion of *subformula* hasn't yet been defined; it will, however, be defined below.

[80] One might wonder why we don't collapse clauses (3.2.a) and (3.2.c) by letting n in the former go to zero. There are several reasons for this. One is that we *read* the formulas of (3.2.a) differently from the way we read the formulas of (3.2.c). See the parenthetical annotation at the right margin

Note that according to clause (.5), when φ is propositional, $[\lambda\, \varphi]$ and φ are distinct 0-place relation terms. In general, the λ-expressions are *not* to be interpreted as denoting functions, but rather as denoting relations.

(**4**) **Remark:** On the Definition of Formula and Propositional Formula. We henceforth mark the distinction between formulas and propositional formulas by using $\varphi, \psi, \chi, \theta$ as metavariables ranging over formulas, and $\varphi^*, \psi^*, \chi^*, \theta^*$ as metavariables ranging over propositional formulas. To be absolutely clear about the distinction, note that some of the clauses in definition (3) include the parenthetical phrase '(propositional)'. This allows us to state the definitions of *formula* and *propositional formula* more efficiently. The clauses containing this parenthetical phrase, namely, (3.2.a), (3.2.c), and (3.3), should all be understood as conjunctions. In each of these clauses, remove the parenthetical phrase entirely to obtain a clause in the definition of *formula*; then remove only the parentheses from all the occurrences of '(propositional)' to obtain a clause in the definition of *propositional formula*. For example, clause (3.3) is short for:

> If φ, ψ are formulas and α any variable, then $(\neg\varphi)$, $(\varphi \rightarrow \psi)$, $\forall\alpha\varphi$, $(\Box\varphi)$, and $(\mathcal{A}\varphi)$ are formulas, and if φ^*, ψ^* are propositional formulas and α any variable, then $(\neg\varphi^*)$, $(\varphi^* \rightarrow \psi^*)$, $\forall\alpha\varphi^*$, $(\Box\varphi^*)$, and $(\mathcal{A}\varphi^*)$ are propositional formulas.

Thus, it should be clear from the definition that the propositional formulas are a subset of the formulas.

Note also that in clause (3.5), we require φ to be propositional, thereby restricting the subclauses introducing complex relation terms: only propositional formulas can appear in such terms. This restriction is achieved by using 'propositional' *without* parentheses in (3.5).

(**5**) **Definition:** Terms. We say that a *term* is any individual term or any n-place relation term ($n \geq 0$). We use τ to range over terms. The simple terms listed in item (1) are terms in virtue of clause (3.1). We say that a term τ is *complex* iff τ is not a simple term. Thus, clauses (3.4), (3.5.a), and (3.5.b) define types of complex terms, namely, *definite descriptions*, *λ-expressions*, and *propositional formulas*, respectively. We sometimes use ρ to range over complex terms. A consequence of our definitions is that every propositional formula is a term.

(**6**) **Definition:** A BNF Definition of the Syntax. We may succintly summarize the essential definitions of the context-free grammar of our language us-

---

of both clauses. A second reason is that (3.2.a) intuitively defines *atomic* exemplification formulas. But (3.2.c) allows *any* 0-place relation term to be a formula, even the complex 0-place relation terms defined in clause (3.5.b). An example of the latter is $Pa\& \neg Qb$. So if we had collapsed (3.2.a) and (3.2.c), then one might be tempted to regard a complex 0-place relation term such as $Pa\& \neg Qb$ as a kind of atomic formula.

ing Backus-Naur Form (BNF). In the BNF definition, we repurpose our Greek metavariables as the names of grammatical categories, as follows:

δ	individual constants
ν	individual variables
Σ^n	n-place relation constants ($n \geq 0$)
Ω^n	n-place relation variables ($n \geq 0$)
α	variables
κ	individual terms
Π^n	n-place relation terms ($n \geq 0$)
φ^*	propositional formulas
φ	formulas
τ	terms

The BNF grammar for our language can now be stated as follows, where the Greek variables are now names of grammatical categories:[81]

	δ	::=	$a_1, a_2, \ldots$
	ν	::=	$x_1, x_2, \ldots$
$(n \geq 0)$	Σ^n	::=	$P^n_1, P^n_2, \ldots$
$(n \geq 0)$	Ω^n	::=	$F^n_1, F^n_2, \ldots$
	α	::=	$\nu \mid \Omega^n\ (n \geq 0)$
	κ	::=	$\delta \mid \nu \mid \iota\nu\varphi$
$(n \geq 1)$	Π^n	::=	$\Sigma^n \mid \Omega^n \mid [\lambda\nu_1 \ldots \nu_n\ \varphi^*]$
	Π^0	::=	$\Sigma^0 \mid \Omega^0 \mid [\lambda\ \varphi^*] \mid \varphi^*$
	φ^*	::=	$\Pi^n\kappa_1 \ldots \kappa_n\ (n \geq 1) \mid \Pi^0 \mid (\neg\varphi^*) \mid (\varphi^* \rightarrow \varphi^*) \mid \forall\alpha\varphi^* \mid$ $(\Box\varphi^*) \mid (\mathcal{A}\varphi^*)$
	φ	::=	$\kappa_1\Pi^1 \mid \varphi^* \mid (\neg\varphi) \mid (\varphi \rightarrow \varphi) \mid \forall\alpha\varphi \mid (\Box\varphi) \mid (\mathcal{A}\varphi)$
	τ	::=	$\kappa \mid \Pi^n\ (n \geq 0)$

Thus, if one defines a fragment of our language by giving a limiting value to n and listing a finite vocabulary of simple terms, the sentences of the resulting grammar can be parsed by any appropriately-configured off-the-shelf parsing engine.

(7) **Definitions:** Notational Conventions. We adopt the following conventions to facilitate readability:

(.1) We often substitute the less formal expressions listed in (1) for their more formal counterparts, and we often drop the superscript indicating the arity of a simple relation term when such terms appear in a formula,

[81] I'm indebted to Uri Nodelman for suggesting a simplification of the BNF grammar I had originally developed. That simplification is included in the definition.

since their arity can always be inferred from the number of individual terms in the formula. Thus, instead of writing $P^1_1 a_1$ and $P^1_2 a_2$, we write Pa and Qb, respectively; instead of $F^1_1 x_1$ and $F^1_2 x_2$, we write Fx and Gy, respectively; instead of $R^2_1 a_2 x_3$, we write Rbz; etc.

(.2) We substitute $p, q, r, \ldots$ for the 0-place relation variables, $F^0_1, F^0_2, \ldots$. If we need 0-place relation constants, we use $p_1, q_2, \ldots$ instead of $P^0_1, P^0_2, \ldots$.

(.3) We omit parentheses in formulas whenever we possibly can, i.e., whenever we can do so without ambiguity, such as in the following circumstances:

- We sometimes add parentheses and square brackets to assist in reading certain formulas and terms. Thus, for example, we write $\imath x(xQ)$ instead of $\imath x xQ$, and $F\imath x(xQ)$ instead of $F\imath x xQ$.
- We almost always drop outer parentheses and assume $\neg$, $\forall$, $\imath$, $\Box$, and $\mathcal{A}$ apply to as little as possible; $\rightarrow$ dominates $\neg$:
  - $\neg Pa \rightarrow Qb$ is short for $((\neg Pa) \rightarrow Qb)$, not $\neg(Pa \rightarrow Qb)$.
  - $\forall x Px \rightarrow Qx$ is to be read as $(\forall x Px) \rightarrow Qx$, not $\forall x(Px \rightarrow Qx)$.
  - $\neg\Box Pa$ is short for $(\neg(\Box Pa))$.
    $\Box Pa \rightarrow Qb$ is short for $(\Box Pa) \rightarrow Qb$, not $\Box(Pa \rightarrow Qb)$
  - $\neg\mathcal{A}Pa$ is short for $(\neg(\mathcal{A}Pa))$.
    $\mathcal{A}Pa \rightarrow Qb$ is short for $(\mathcal{A}Pa) \rightarrow Qb$, not $\mathcal{A}(Pa \rightarrow Qb)$.

(.4) We employ the usual definitions of (a) φ *and* ψ, (b) φ *or* ψ, (c) φ *if and only if* ψ, (d) *there exists an* α *such that* φ, and (e) *possibly* φ:

(a) $\varphi \,\&\, \psi \;=_{df}\; \neg(\varphi \rightarrow \neg\psi)$

(b) $\varphi \vee \psi \;=_{df}\; \neg\varphi \rightarrow \psi$

(c) $\varphi \equiv \psi \;=_{df}\; (\varphi \rightarrow \psi) \,\&\, (\psi \rightarrow \varphi)$

(d) $\exists\alpha\varphi \;=_{df}\; \neg\forall\alpha\neg\varphi$

(e) $\Diamond\varphi \;=_{df}\; \neg\Box\neg\varphi$

(.5) $\rightarrow$ and $\equiv$ dominate & and $\vee$:

$\neg\varphi \,\&\, \psi \rightarrow \chi$ is short for $(\neg\varphi \,\&\, \psi) \rightarrow \chi$, not $\neg\varphi \,\&\, (\psi \rightarrow \chi)$

$\varphi \vee \psi \equiv \psi \vee \varphi$ is short for $(\varphi \vee \psi) \equiv (\psi \vee \varphi)$

Given the definitions of & and $\vee$ in (.3) and (.4), it should be clear that we are preserving a classical understanding of these connectives.

(**8**) **Definition:** Subformula. Where φ is any formula, we define a *subformula* of φ recursively as follows:

(.1) φ is a subformula of φ.

(.2) If $\neg\psi$, $\forall\alpha\psi$, $\Box\psi$, or $\mathcal{A}\psi$ is a subformula of φ, then ψ is a subformula of φ.

(.3) If $\psi \rightarrow \chi$ is a subformula of φ, then ψ is a subformula of φ and χ is a subformula of φ

(.4) If $[\lambda\, \psi^*]$ is a subformula of φ, then ψ^* is a subformula of φ.

Given this definition, we may say that ψ is a *proper* subformula of φ just in case ψ is a subformula of φ and ψ is not φ.

One interesting feature of our definition of *subformula* is that it doesn't count the formula Gx as a subformula of the formulas $F\imath xGx$ or $[\lambda x\, \neg Gx]a$. Indeed, the definition generally doesn't count the formula φ in the term $\imath\nu\varphi$, or the formula φ^* in the term $[\lambda\nu_1 \ldots \nu_n\, \varphi^*]$ $(n \geq 1)$, as subformulas of any formula in which these terms occur. Consequently, if φ contains encoding subformulas, then the description $\imath\nu\varphi$ *may appear* in propositional formulas! For example, though the formula xQ contains an encoding subformula (namely, itself), the formula $F\imath x(xQ)$ qualifies as a propositional formula. This is an exemplification formula with the complex term $\imath x(xQ)$ appearing the argument to the relation term F. (8) doesn't define the notion *subformula of* for the term $\imath x(xQ)$ and so the formula xQ doesn't qualify as a subformula of $F\imath x(xQ)$. Consequently, the expressions $[\lambda F\imath x(xQ)]$ and $[\lambda y R y\imath x(xQ)]$ are perfectly well-defined 0-place and 1-place λ-expressions, respectively. The reader might wish to consider how the semantics we developed in Chapter 5 provides truth conditions for (a) propositional formulas that contain descriptions with encoding formulas, such as $\imath x(xQ)$, and (b) propositional formulas such as $[\lambda y\, Ry\imath x(xQ)]z$, in which the λ-operator λy governs a propositional formula containing a term that includes encoding formulas.

Our definition of *subformula* of has some other important consequences, notably:

> **Metatheorem** ⟨7.1⟩
> If ψ is a subformula of χ and χ is a subformula of φ, then ψ is a subformula of φ.

> **Metatheorem** ⟨7.2⟩
> φ is a propositional formula if and only if no encoding formulas are subformulas of φ.

> **Metatheorem** ⟨7.3⟩
> φ and ψ are subformulas of $\varphi \,\&\, \psi$, $\varphi \vee \psi$, and $\varphi \equiv \psi$.

These metatheorems are proved or left as exercises in the Appendix to this chapter (see Part IV). Given Metatheorem ⟨7.2⟩ and clause (3.5), we can see

that only formulas φ without encoding subformulas may appear in terms that denote relations.

(**9**) **Definitions:** Operator Scope and Free (Bound) Occurrences of Variables. We first define the *scope* of an occurrence of the formula- and term-building operators as follows:

(.1) The formulas $\neg\psi$, $\Box\psi$, and $\mathcal{A}\psi$ are the scope of the occurrence of the operators $\neg$, $\Box$ and $\mathcal{A}$ in their respective formulas. The formula $\psi \rightarrow \chi$ is the scope of the occurrence of the operator $\rightarrow$.

(.2) The formula $\forall\beta\psi$ is the scope of the left-most occurrence of the operator $\forall\beta$ in that formula. (We sometimes call ψ the proper scope or *matrix* of $\forall\beta$ in $\forall\beta\psi$.)

(.3) The term $\iota\nu\psi$ is the scope of the left-most occurrence of the operator $\iota\nu$ in that term. (We sometimes call ψ the proper scope or matrix of $\iota\nu$ in $\iota\nu\psi$.)

(.4) The term $[\lambda\nu_1 \ldots \nu_n\, \psi^*]$ $(n \geq 0)$ is the scope of the left-most occurrence of the operator $\lambda\nu_1 \ldots \nu_n$ in that term. (We sometimes call ψ^* the proper scope or matrix of $\lambda\nu_1 \ldots \nu_n$ in $[\lambda\nu_1 \ldots \nu_n\, \psi^*]$.)

Now to define *free* and *bound* occurrences of a variable within a formula or term, we first say that $\forall\beta$ is a variable-binding operator *for* β, that $\iota\nu$ is a variable-binding operator *for* ν, and that $\lambda\nu_1 \ldots \nu_n$ is a variable-binding operator *for* $\nu_1, \ldots, \nu_n$. We then say, for any variable α and any formula φ (or any complex term τ):

(.5) An occurrence of α in φ (or τ) within the scope of an occurrence of a variable-binding operator for α is *bound*; otherwise, the occurrence is *free*.

Finally, we say:

(.6) Those occurrences of β that are free in ψ are *bound by* the left-most occurrence of $\forall\beta$ in $\forall\beta\psi$, as is the occurrence of β in that occurrence of $\forall\beta$; those occurrences of ν that are free in ψ are *bound by* the left-most occurrence of $\iota\nu$ in $\iota\nu\psi$, as is the occurrence of ν in that occurrence of $\iota\nu$; and those occurrences of ν_i $(1 \leq i \leq n)$ that are free in ψ^* are *bound by* the left-most occurrence of $\lambda\nu_1 \ldots \nu_n$ in $[\lambda\nu_1 \ldots \nu_n\, \psi^*]$, as are the occurrences of $\nu_1, \ldots, \nu_n$ in that occurrence of $\lambda\nu_1 \ldots \nu_n$.

We henceforth say that the variable α *occurs free* or *is free* in formula φ or term τ if and only if at least one occurrence of α in φ or τ is free. Note that we don't need to define when an occurrence of a variable is free in the 0-place term φ^*,

i.e., propositional formula φ^*. Since these expressions constitute a subset of the formulas, this case is covered by the general definition of free occurrence of a variable in a formula.

(**10**) **Remark:** Open and Closed Formulas and Terms. In the foregoing, we have rigorously distinguished between terms and formulas, though we've defined some expressions to be both terms and formulas. Intuitively, a *term* is an expression that may have a denotation, relative to some choice of values for the free variables it may contain. By contrast, a *formula* is, intuitively, an expression that is assertible and has truth conditions, relative to some choice of values for any free variables it may contain. A formula φ is *closed* if no variable occurs free in φ; otherwise φ is *open*. A formula φ is a *sentence* iff φ is a closed formula. We say further that a term τ is *closed* if no variable occurs free in τ; otherwise, τ is *open*. We may sometimes refer to open complex terms, such as $\imath xRxy$ and $[\lambda x\, Rxy]$, as *complex variables*. The former example is a complex individual variable, since for each choice for y, it may denote a different individual; the latter is a complex 1-place relation variable since, for each choice of y, it may denote a a different 1-place relation.

7.2 Definitions for Objects and Relations

(**11**) **Term Definitions:** Ordinary vs Abstract (i.e., Logical) Objects. We define *being ordinary* ('$O!$') as a new 1-place relation term:

(.1) $O! =_{df} [\lambda x \Diamond E!x]$

Strictly speaking, then, being ordinary is: being an x such that possibly, x exemplifies concreteness. In other words, being ordinary is being possibly concrete.

On the other hand, *being abstract* or *being logical*, written $A!$, is defined as:

(.2) $A! =_{df} [\lambda x \neg\Diamond E!x]$

So being abstract (or being logical) is being an x such that it is not possible that x exemplify concreteness, or more simply, being an x that couldn't possibly exemplify concreteness, or even more simply, not possibly being concrete.

(**12**) **Term Definition:** The Identity$_E$ Relation. We define the two-place relation *being identical*$_E$ ('$=_E$') as: being an individual x and an individual y such that x exemplifies being ordinary, y exemplifies being ordinary, and necessarily, x and y exemplify the same properties:

$=_E =_{df} [\lambda xy\, O!x \,\&\, O!y \,\&\, \Box\forall F(Fx \equiv Fy)]$

Note that the definiens is a *bona fide* λ-expression (modulo the less formal variables) because there are no encoding subformulas in the matrix formula. Since the definiens is a well-defined 2-place relation term, so is the definiendum and thus, exemplification formulas like $=_E xy$ are well-formed.

(**13**) **Definition:** Identity$_E$ Infix Notation. We henceforth use the following notation for formulas containing the relation term $=_E$:

$$x =_E y \ =_{df} \ =_E xy$$

(**14**) **Remark:** Nested λ-expressions. It is of interest that the notion of a *haecceity* of individual a, namely, $[\lambda x\, x =_E a]$, which we discussed in earlier chapters, is now defined in a rather complex way, involving nested λ-expressions. For by the infix notation we've just defined,

$$[\lambda x\, x =_E a]$$

unpacks, by definition (13), to:

$$[\lambda x \ =_E xa]$$

which in turn expands, by definition (12) of $=_E$, to:

$$[\lambda x\, [\lambda xy\, O!x \,\&\, O!y \,\&\, \Box\forall F(Fx \equiv Fy)]xa]$$

This, in turn, expands by definition (11.1) of $O!$, to:

$$[\lambda x\, [\lambda xy\, [\lambda x \Diamond E!x]x \,\&\, [\lambda x \Diamond E!x]y \,\&\, \Box\forall F(Fx \equiv Fy)]xa]$$

This expands, by definition (7.4.e) of $\Diamond$, to:

$$[\lambda x\, [\lambda xy\, [\lambda x \neg\Box\neg E!x]x \,\&\, [\lambda x \neg\Box\neg E!x]y \,\&\, \Box\forall F(Fx \equiv Fy)]xa]$$

Of course, this λ-expression is still not in primitive notation, since the definitions of & and $\equiv$ allow us to expand the expression even further. But enough has been said to alert the reader to the many layers of definition that are accumulating.

(**15**) **Definition:** Identity For Individuals. The symbol '=' for identity is not among the primitive expressions of our language. It can, however, be defined by cases, namely, for the two kinds of entities: individuals and n-place relations ($n \geq 0$). In turn, the definition of '=' for n-place relations will itself be given by cases (i.e., for properties, relations, and propositions) in (16). Note that though the following definitions allow us to show that identity is reflexive, the full power of identity as an equivalence condition is guaranteed by axiom (25).

But we first define identity for individuals. We say that $x = y$ iff either $x =_E y$ or x and y are both abstract and necessarily encode the same properties:

$$x = y \;=_{df}\; x =_E y \vee (A!x \,\&\, A!y \,\&\, \Box\forall F(xF \equiv yF))$$

Note that the definiens has encoding subformulas. Consequently, the expression $[\lambda xy\, x = y]$ is not a well-formed λ-expression; the matrix formula $x = y$, once expanded into primitive notation, contains encoding subformulas. (The defined formula '$x = y$' is therefore unlike, and is to be rigorously distinguished from, the defined term '$=_E$'.) Similarly, the expression $[\lambda x\, x = y]$ is not well-formed. This forestalls the McMichael/Boolos paradox discussed in Section 2.1. Moreover, the argument for the paradox fails to get any purchase on properties of the form $[\lambda x\, x =_E y]$.

(**16**) **Definitions:** Relation Identity. We define identity for properties, relations, and propositions as follows: (.1) properties F^1 and G^1 are identical iff necessarily, F^1 and G^1 are encoded by the same objects; (.2) relations F^n and G^n are identical iff for each way of plugging n–1 objects into the corresponding argument places of F^n and G^n, the resulting 1-place properties are identical; (.3) propositions F^0 and G^0 are identical iff *being such that* F^0 is identical to *being such that* G^0. Formally:

(.1) $F^1 = G^1 \;=_{df}\; \Box\forall x(xF^1 \equiv xG^1)$

(.2) $F^n = G^n \;=_{df}\;$ (where $n \geq 2$)

$$\begin{aligned}\forall x_1 \ldots\, \forall x_{n-1}([\lambda y\, F^n y x_1 \ldots x_{n-1}] = [\lambda y\, G^n y x_1 \ldots x_{n-1}] \,\&\\ [\lambda y\, F^n x_1 y x_2 \ldots x_{n-1}] = [\lambda y\, G^n x_1 y x_2 \ldots x_{n-1}] \,\&\ldots\&\\ [\lambda y\, F^n x_1 \ldots x_{n-1} y] = [\lambda y\, G^n x_1 \ldots x_{n-1} y])\end{aligned}$$

(.3) $F^0 = G^0 \;=_{df}\; [\lambda y\, F^0] = [\lambda y\, G^0]$, i.e., using our conventions:
$p = q \;=_{df}\; [\lambda y\, p] = [\lambda y\, q]$

The last two definitions reduce the identity of relations and propositions, respectively, to the identity of properties. The simplest case of (.2) is where $n = 2$, which asserts:

$$F^2 = G^2 \;=_{df}\; \forall x_1([\lambda y\, F^2 y x_1] = [\lambda y\, G^2 y x_1] \,\&\, [\lambda y\, F^2 x_1 y] = [\lambda y\, G^2 x_1 y])$$

This asserts that F^2 and G^2 are identical if and only if for any object x_1, both (a) the property *bearing* F^2 *to* x_1 is identical to the property *bearing* G^2 *to* x_1 and (b) the property *being an object to which* x_1 *bears* F^2 is identical to the property *being an object to which* x_1 *bears* G^2. By (.1), the properties asserted to be identical in (a) and (b) have to be necessarily encoded by the same objects. Hence, F^2 and G^2 are identical if and only if, for every x_1, (a) there couldn't possibly be an object that encodes $[\lambda y\, F^2 y x_1]$ and fails to encode $[\lambda y\, G^2 y x_1]$ (or vice versa), and (b) there couldn't possibly be an object that encodes $[\lambda y\, F^2 x_1 y]$ and fails to encode $[\lambda y\, G^2 x_1 y]$ (or vice versa).

(**17**) **Remark:** Variables in Definitions. Note that we formulated definitions (11) – (16) using the object-language variables listed in the 'Less Formal' column of item (1.1) and (1.2). Specifically, we always try to use $x, y, z, \ldots$ instead of $x_1, x_2, x_3, \ldots$, and use $F, G, H, \ldots$ instead of $F^1_1, F^1_2, F^1_3, \ldots$, etc. The reason for this convention should be clear: the definiens of (11.1), $[\lambda x \Diamond E!x]$, is easier to read than $[\lambda x_1 \Diamond E!x_1]$, and the definiens of (12),

$$[\lambda xy\, O!x \,\&\, O!y \,\&\, \Box\forall F(Fx \equiv Fy)]$$

is easier to read than:

$$[\lambda x_1 x_2\, O!x_1 \,\&\, O!x_2 \,\&\, \Box\forall F^1_1(F^1_1 x_1 \equiv F^1_1 x_2)]$$

Nevertheless, we should remember that the definienda, '$O!$' and '$=_E$', are particular symbols being introduced and their definientia are particular λ-expressions in the object language. The 'less formal' variables $x, y, z \ldots$ are merely notational shorthand for the variables that are officially part of the language.

This observation applies to both definiendum and definiens in definitions such as (16.1). The definiendum $F^1 = G^1$ and its definiens $\Box\forall x(xF^1 \equiv xG^1)$ are easier to read, respectively, than $F^1_1 = F^1_2$ and $\Box\forall x_1(x_1F^1_1 \equiv x_1F^1_2)$. The improvement in readability is obvious and should justify the deployment of 'less formal' variables. Indeed, that is why we shall henceforth suppose that the definiendum can be written simply as $F = G$ and that the definiens can be written as $\Box\forall x(xF \equiv xG)$. But, officially, the definiendum and definiens use variables that are part of our specified language.

(**18**) **Definitions:** Infix Notation for Negated Identities. We henceforth use the following infix notation for the negation of formulas involving the identity symbol:

> $\alpha \neq \beta \;=_{df}\; \neg(\alpha = \beta)$,
> where α, β are, respectively, either both individual variables or both n-place relation variables (for some $n \geq 0$).

(**19**) **Remark:** A Note about Definitions. In this work, we regard definitions not as metalinguistic abbreviations of the object language but rather as conventions for extending the object language with new terms, formulas and axioms. Our main reason for understanding definitions in this way is so that theorems involving defined notions become genuine philosophical statements of the object language rather than metaphilosophical statements of the metalanguage. Though we shall postpone a full discussion of the theory of definitions to Section 9.11 of Chapter 9, it is important to make a few observations about the way definitions are constructed and understood.

We shall be using two kinds of definitions to extend our system: *term definitions* to introduce new terms, and *formula definitions* to introduce formulas

containing new syncategorematic expressions. For example, the definitions in items (11) and (12) are term definitions: $O!$, $A!$, and $=_E$ are new terms introduced into the language. In these particular examples, the new terms are *relation constants* since the definientia have no free variables. However, free variables may occur in definitions: we may use complex variables as definientia to introduce *function terms*. For example, we might define the negation of property F as follows:

$$\overline{F} =_{df} [\lambda x \neg Fx]$$

In this definition, the variable F occurs free in both definiendum and definiens, and the definiens $[\lambda x \neg Fx]$ is a complex variable that serves to define the new function term $\overline{F}$. For each property that F takes as value, $\overline{F}$ denotes the negation of that property.

Indeed, one can uniformly substitute appropriate property terms for the free variable F to produce instances of the definition. Consider the the open property term $[\lambda y\ Gy \,\&\, Hy]$, in which G and H are free variables. We can substitute $[\lambda y\ Gy \,\&\, Hy]$ uniformly for F to produce the following instance of the definition:

$$\overline{[\lambda y\ Gy \,\&\, Hy]} =_{df} [\lambda x \neg [\lambda y\ Gy \,\&\, Hy]x]$$

The property term $[\lambda y\ Gy \,\&\, Hy]$ is an appropriate substitution for F because it contains no free occurrences of the variable x; if a property term ρ contains a free occurrence of x, the occurrence of x would become bound by the operator λx if ρ were substituted for F in $[\lambda x \neg Fx]$, thereby changing the meaning of the definition. The notion of appropriateness will become clear when, in item (24), we define the conditions under which a term τ is *substitutable for* the variable α in a term or formula.[82]

We shall also have occasion to formulate definitions of new individual terms. In the simplest case, we define a new individual term by employing a closed definite description (i.e., no free variables) that provably has a denotation. For example, since it will be provable that the abstract object that encodes all and only self-identical properties exists, we might introduce the individual constant $\boldsymbol{a_V}$ as follows:

$$\boldsymbol{a_V} =_{df} \imath x(A!x \,\&\, \forall F(xF \equiv F = F))$$

Free variables are also allowed in definitions of individual terms: open definite descriptions (i.e., those having free variables) are complex variables that can be

[82] That definition makes it clear that even the term $[\lambda x\ Gx \,\&\, Hx]$ can be substituted for F in the definition to produce the instance:

$$\overline{[\lambda x\ Gx \,\&\, Hx]} =_{df} [\lambda x \neg [\lambda x\ Gx \,\&\, Hx]x]$$

The occurrences of x in $[\lambda x Gx \& Hx]$ are all bound and so the none get captured when we substitute this expression for F in $[\lambda x \neg Fx]$.

used to define a function term. For example, since it will be provable that for every property G, the abstract object that encodes all and only the properties identical to G exists, we might introduce the function term $\boldsymbol{a}_G$ as follows:

$$\boldsymbol{a}_G \;=_{df}\; \iota x(A!x \,\&\, \forall F(xF \equiv F = G))$$

In this definition, the variable G occurs free in both definiendum and definiens, and the definiens is a complex variable that serves to define the new function term $\boldsymbol{a}_G$. For each property that G takes as value, the function term $\boldsymbol{a}_G$ denotes the abstract object that encodes just the properties identical to G.

Again, one can uniformly substitute appropriate property terms for the free variable G to produce instances of the definition. For example, we can substitute the term $[\lambda y\, \neg Hy]$ uniformly for G to produce the following instance of the definition:

$$\boldsymbol{a}_{[\lambda y\, \neg Hy]} \;=_{df}\; \iota x(A!x \,\&\, \forall F(xF \equiv F = [\lambda y\, \neg Hy]))$$

Indeed, we can produce an instance of the definition by substituting a defined term for G:

$$\boldsymbol{a}_{\overline{H}} \;=_{df}\; \iota x(A!x \,\&\, \forall F(xF \equiv F = \overline{H}))$$

Although it is not appropriate to substitute the variable F for G in the definition of $\boldsymbol{a}_G$ (since F would become bound by the quantifier $\forall F$ in the definiens), our theory considers the definition to be unchanged if both the bound and free variables in the definiendum and definiens are appropriately changed, for example, as follows:

$$\boldsymbol{a}_F \;=_{df}\; \iota x(A!x \,\&\, \forall G(xG \equiv G = F))$$

The justification for understanding the definition in this way is given in item (204).

Although we intend a classical understanding of the inferential role of term definitions, the subtleties involved in our language make it incumbent upon us to provide a thorough discussion of this role, and this we do in item (202). For now, the main points to remember are that in any properly formed term definition of the form $\tau =_{df} \tau'$:

- the definiendum and definiens both have the same free variables, if any;
- the definition is independent of the choice of any free and bound variables; any other appropriate variables could have been used instead;
- it must be provable that $\exists\beta(\beta = \tau')$, i.e., a term τ' can be used to define a new term τ only if it provably denotes something;

- whenever appropriate terms of the right type are uniformly substituted for any free variables in the definition, the result is an instance of the definition;
- every well-formed instance of the definition becomes assertible as a simple identity of the form $\tau = \tau'$ (202.2); and
- for any instance of the definition, the defieniendum τ and definiens τ' may be substituted for one another in any context (202.3).

Although it won't become completely clear why these facts govern the inferential role of definitions until (202) and (204), they assure us that expressions introduced into our language via term definitions are logically well-behaved.

Term definitions are to be contrasted with *formula definitions*. There are two types of formula definitions: those that use metavariables and those that use object language variables. Examples of the first kind are items (7.4.a), (7.4.d), and (7.4.e):

$$\varphi \,\&\, \psi \;=_{df}\; \neg(\varphi \rightarrow \neg\psi) \tag{7.4.a}$$

$$\exists\alpha\varphi \;=_{df}\; \neg\forall\alpha\neg\varphi \tag{7.4.d}$$

$$\Diamond\varphi \;=_{df}\; \neg\Box\neg\varphi \tag{7.4.e}$$

In these cases, new syncategorematic expressions &, $\exists$, $\Diamond$ are being introduced into the language by way of formulas. These symbols are akin to $\neg$, $\rightarrow$, $\forall$, and $\Box$. Syncategorematic expressions, unlike terms, don't denote anything, though the formulas in which they occur have truth conditions. Any uniform substitution of object-language formulas for the Greek metavariables results in an instance of the above definitions.

The second kind of formula definition is similar to the first except that it uses object language variables instead of metavariables. Examples are items (15) and (16.1):

$$x = y \;=_{df}\; x =_E y \vee (A!x \,\&\, A!y \,\&\, \Box\forall F(xF \equiv yF)) \tag{15}$$

$$F = G \;=_{df}\; \Box\forall x(xF \equiv xG) \tag{16.1}$$

Note that this second kind of formula definition also introduces new syncategorematic expressions. In (15) and (16.1), the identity symbol '=' introduced is not a new 2-place relation term. Rather, '=' is a syncategorematic expression that functions as a formula-forming binary operator on variables of the same type, though the definition of the formula that results depends on the type of variable on which '=' operates.

We'll see additional examples, such as the following, in later chapters:

$$\mathit{ExtensionOf}(x, G) \;=_{df}\; A!x \,\&\, \forall F(xF \equiv \forall z(Fz \equiv Gz)) \tag{229.1}$$

$$ClassOf(x, G) =_{df} ExtensionOf(x, G) \tag{229.2}$$

Thus both *ExtensionOf* and *ClassOf* become binary formula-forming operators that takes an individual variable as its first argument and a property variable as its second. We can produce instances of these definitions when we substitute appropriate individual terms for x and appropriate property terms for G.

A full discussion of the conventions for, and inferential role of, formula definitions is postponed until items (203) and (204), when we have proved enough theorems to justify our practice. The main points to remember now are that in a properly formed formula definition of the form $\varphi =_{df} \psi$:

- the definiendum and definiens both have the same free metavariables or the same free object-language variables (depending on the kind of formula definition);
- the definition is independent of the choice of metavariables and independent of the choice of bound and free object-language variables; any other appropriate (meta)variables could have been used instead;
- whenever formulas are uniformly substituted for the free metavariables in the definition, or appropriate terms are uniformly substituted for the free object language variables in the definition, the result is an instance of the definition;
- the definition subsequently becomes assertible as a necessary equivalence of the form $\Box\varphi \equiv \psi$;[83] and
- for any instance of the definition, the definiendum φ and definiens ψ can be substituted for one another within any formula or term.

These facts ensure that new expressions introduced into our language via formula definitions are logically well-behaved.

Although formula definitions introduce new syncategorematic expressions, the expressions introduced nevertheless typically provide an analysis of the meaning of a logically or philosophically significant word or phrase of English in terms of the (primitive) notions represented by the primitive expressions of our language. In the above cases: (7.4.a) analyzes 'and', (7.4.d) analyzes 'there exists', and (7.4.e) analyzes 'possibly'. Similarly, definition (15) of '=' offers an analysis of the English expression 'is identical to' as used in claims such as "Hesperus is identical to Phosphorus". Definition (16.1), which provides a

[83] Indeed, something stronger is warranted. When we define *theoremhood* in Chapter 9 and distinguish a special class of *modally strict* theorems that are derivable without appealing to an axiom or premise that can't be necessitated, then we can say: a formula definition of the form $\varphi =_{df} \psi$ becomes assertible as a modally strict theorem of the form $\varphi \equiv \psi$. See the Simple Rule of Equivalence by Definition in item (203.2).

separate case of the definition of '=', offers an analysis of the English phrase 'is the same as' as used in such claims as "*being a brother* is the same as *being a male sibling*" and "*being a circle* is the same as *being a closed plane figure every point of which lies equidistant from some given point*". And so on for the other examples of formula definitions.

Of course, the hope is that such definitions provide insightful analyses that demonstrate the power of the primitive notions of the language. For example, the definition of $F = G$ is part of our *theory* of identity: it (a) analyzes the identity of properties in terms of the primitives of our language, (b) provides a precise formulation of a notion (i.e., property identity) thought to be mysterious (e.g., by Quine), (c) allows that properties may be distinct even if necessarily exemplified by the same objects, and (d) shows that the identity conditions of properties are extensional (semantically, the definition requires that properties F and G have the same encoding extension at every semantically-primitive possible world). We need not pursue these facts, since they have been discussed elsewhere in this work.

It is interesting to observe that the above distinction between term definitions and formula definitions is not mutually exclusive: in the special case where the definiens is a propositional formula, the definition is both a term definition as well as a formula definition. Such definitions have features of both formula and term definitions. Consider the following two examples:

$$G \Rightarrow F \;=_{df}\; \Box\forall x(Gx \rightarrow Fx) \tag{321.1}$$

$$p \,\&\, q \;=_{df}\; \neg(p \rightarrow \neg q) \qquad \text{instance of (7.4.a)}$$

$$Px \,\&\, Qx \;=_{df}\; \neg(Px \rightarrow \neg Qx) \qquad \text{instance of (7.4.a)}$$

At first glance, these definitions appear to introduce $\Rightarrow$ and & as syncategorematic expressions by way of the formulas $G \Rightarrow F$ and $Px \,\&\, Qx$. In this respect, these definitions may be considered formula definitions and so the definitions become assertible as equivalences.

However, in each case, the definiens is a propositional formula, and so by convention, the definiendum is as well. The definition of $G \Rightarrow F$ can be understood as introducing the special new binary function term that denotes, relative to any properties G and F, the proposition (i.e., 0-place relation) that necessarily, everything that exemplifies G exemplifies F. (Indeed, we could make the definiendum look more like a function term by writing it as $\Rightarrow_{G,F}$.) The definition of $p \,\&\, q$ shows that in the special case where & conjoins propositional formulas, it is not just a syncategorimatic expression but also a binary function symbol that denotes a proposition relative to the values of its arguments. (Again, we could make the definiendum look more like a function term by writing it as $\&_{p,q}$.) In this respect, all of these definitions or instances of def-

initions may be considered term definitions. On the basis of such definitions, we may assert the identities:

$$(G \Rightarrow F) = \Box\forall x(Gx \rightarrow Fx)$$

$$(p \,\&\, q) = \neg(p \rightarrow \neg q)$$

$$(Px \,\&\, Qx) = \neg(Px \rightarrow \neg Qx)$$

In general, when a definiens is a propositional formula ψ^*, both the definiendum φ^* and definiens are terms and the definition becomes assertible as a simple identity of the form $\varphi^* = \psi^*$.

In the deductive system described in Chapter 9, identities of the form $\varphi^* = \psi^*$ will logically imply equivalences of the form $\varphi^* \equiv \psi^*$, but not vice versa. Since definitions involving propositional formulas yield such identities, we'll suppose that that they are governed by both our conventions for term definitions and for formula definitions. We are free to draw inferences that conform to either conventions. However, we'll *label* such definitions as **Definitions** and reserve the label **Term Definition** when the expressions flanking the $=_{df}$ sign are terms that aren't formulas. So, we'll continue to use separate labels despite the fact the categories are not mutually exclusive.

Finally, the above definitions and remarks, together with the full theory of definitions described below in Section 9.11, provide a precise syntactic characterization of the formalism and thus forestall the concerns that Gödel (1944, 120) raised in this regard about Whitehead and Russell's *Principia Mathematica*:

> It is to be regretted that this first comprehensive and thorough-going presentation of a mathematical logic and the derivation of mathematics from it [is] so greatly lacking in formal precision in the foundations (contained in *1–*21 of *Principia*) that it presents in this respect a considerable step backwards as compared with Frege. What is missing, above all, is a precise statement of the syntax of the formalism. Syntactical considerations are omitted even in cases where they are necessary for the cogency of the proofs, in particular in connection with the "incomplete symbols".

It is important to have taken some pains in the above development to ensure that such a criticism doesn't apply to the present effort.