

11

Second-Order Modal Logic

1.	Introduction.....	2
2.	Second-Order Considerations	2
3.	Monadic Plurals	4
4.	Syntax for Modal Logic with Plurals – ML+P.....	4
	1. Logical Vocabulary	5
	2. Non-Logical Vocabulary	5
	3. Rules of Formation.....	6
5.	Rules of Derivation	7
	1. Notational Conventions	7
	2. Quantifier Rules.....	7
	3. Identity Rules.....	8
	4. Further Second-Order Rules.....	9
6.	How to Read Second-Order Quantifiers in ML+P.....	10
7.	Lambda-Conversion and Scoped-Terms.....	10
8.	Scoped-Predicates.....	11
9.	Scoped-Predicates and Actuality.....	12
10.	Why the Second-Order Quantifier Rules Require Constants.....	15

1. Introduction

In an earlier chapter, we saw that certain sentences of English can be formalized using the actuality operator. We also saw that, in cases in which this operator is a term-modifying adverb, the formalization in terms of the actuality operator is equivalent to a Russell-like formalization in terms of scoped singular-terms.

In the present chapter, we extend the Russell-style analysis so that it can also be applied to the actuality operator used as a sentential adverb. In order to accomplish this analysis, we formulate a logic of *monadic plurals*, which we formalize as a *second-order* logic.

2. Second-Order Considerations

We begin by briefly examining the natural-language underpinnings of second-order logic. Consider the following sentence.

Kay is eating the birthday cake	$E[K,B]$
---------------------------------	----------

If we examine this sentence from the viewpoint of first-order logic, we only consider quantifying "into" singular-term position. Results of singular-term quantification include sentences such as:

Kay is eating something	$\exists x E[K,x]$
someone is eating the birthday cake	$\exists x E[x,B]$

On the other hand, first-order logic does not permit us to quantify into the verb position to obtain a sentence like

✖ Kay is something the birthday cake	$\exists X X[K,B]$
--------------------------------------	--------------------

since the latter expression is ill-formed in first-order logic.

The above phrase is also ill-formed according to English, which doesn't allow *simple* quantification into verb position either. However, English does allow gerundial-quantification. For these purposes, we take a *gerund* to be a sort of *nominalized-verb* – examples of which include:

eating walking running swimming

Any one of these expressions is a grammatically admissible answer to the question

what is Kay doing?

although none is a grammatically (semantically) admissible answer to:

what is Kay eating?

Let us examine the difference between the two questions. On the one hand, the question

what are you eating?

expects an answer of the form

I am eating _____

whose instances include the following sorts of sentences.

I am eating an apple

I am eating the apple you gave me

On the other hand, the question

what are you doing?

does not expect an answer of the following form.¹

I am doing _____

Rather, it expects an answer of the form

I am _____ ing _____

whose instances include the following sorts of sentences.

I am eat ing ...

I am eat ing an apple

Now, as a general rule in English, if one can place the word 'what' into a grammatical location in a sentence (properly transformed), then one can also place the word 'something' there (properly re-transformed). Thus, we obtain the following questions by replacing 'what' by 'something' and putting the expression back into its "deep structure" location.

are you eating something?

are you doing something?

In spite of the huge *ontological* difference between the "thing" you are eating and the "thing" you are doing, competent English speakers have no problem answering these questions. Indeed, the answer to both questions might be the very same, syntactically at least.

yes, I am eating an apple

On the other hand, it is quite clear that, whereas there is a *thing* (i.e., individual entity) that you are *eating*, there is no *thing* (i.e., individual entity) that you are *doing*.

Now, we can capture this ontological difference between doing *something* and eating *something* by distinguishing two forms of quantification. In particular, whereas

eating *something* is an instance of first-order quantification;
doing *something* is an instance of second-order quantification.

In order to achieve this, second-order logic expands the notion of quantification, so that one can quantify "into" categories besides singular-terms. In addition to being able to quantify into noun-phrase position, one can also quantify into predicate position; moreover, one can do this for any predicate degree, including zero-degree.

¹ Of course you might be "doing" errands, or "doing" laundry, or "doing" taxes, but here the verb 'to-do' is categorially quite different; here, 'do' is a functor that converts a noun phrase into a verb phrase. For example, rather than say 'I am erranding', one says 'I am doing errands'.

3. Monadic Plurals

Briefly, a monadic plural is the sort of thing that a plural noun phrase (of the usual sort) denotes. In elementary logic, monadic plural nouns are represented by monadic (1-place) predicates – hence the name ‘monadic plural’. Some plural expressions can be adequately translated into the formal language of elementary (i.e., first-order) predicate logic; some cannot. For example, contrast the following pairs.

all buffalo are four-legged	buffalo are rare
my brothers all live in Boston ²	my brothers all live together
no evil men are virtuous	evil men are many
the protestors all carried signs	the protestors surrounded the building

Whereas a *monadic* plural noun phrase refers to a "bunch" of individual things *all at once*, a *dyadic* plural noun phrase refers to a "bunch" of pairs/couples all at once. In contrast to monadic plural noun phrases, which are quite common, dyadic plural noun phrases are considerably less common. The following is a simple example. (Imagine a health education professional talking.)

we had many **couples** at last night’s meeting;
we divided **them** into groups for separate discussions.

Here, we understood ‘couples’ to refer to *pairs* of people related to one another in a contextually salient way.

Since dyadic plurals do not pose any special problems over and above the problems already posed by monadic plurals, we concentrate our attention on monadic plurals, and restrict our formal syntax accordingly.

4. Syntax for Modal Logic with Plurals – ML+P

In order to produce a formal language capable of expressing *monadic* plurals, we employ a comparatively tame version of second-order logic with modal operators. In order to produce this formal language, which we call ML+P, we begin with our formal language for QML+A, and we append additional, more complex, syntactic constructions, as well as their associated inference rules. Notice that, although we officially include the actuality operator in this language, it is dispensable.

The official syntax of ML+P, is given as follows.

² The expression ‘brothers’ is an example of a *relational noun*, which gives rise to a monadic plural noun phrase when supplied with an argument in genitive case.

1. Logical Vocabulary

Name	Instances	Category
individual variables	x, y, z , etc.	$V_0 [=_{df} N(v)]$
individual constants	a, b, c , etc.	N
monadic-predicate variables	X, Y, Z , etc.	$V_1 [=_{df} (N \rightarrow S)(v)]$
monadic-predicate constants	A, B, C , etc.	$N \rightarrow S$
SL connectives	$\sim, \rightarrow, \leftrightarrow, \&, \vee$	$S^k \rightarrow S$
quantifiers	\forall, \exists	$(V_0 \times S) \rightarrow S$
		$(V_1 \times S) \rightarrow S$
description operator	ι	$(V_0 \times S) \rightarrow N$
identity sign	$=$	$N^2 \rightarrow S$;
		$(N \rightarrow S)^2 \rightarrow S$
modal connectives	\Box, \Diamond	$S \rightarrow S$
(flexible) actuality operator	\bigcirc	$N \rightarrow N$
		$S \rightarrow S$
scoped-term/predicate operator	$(_ / _)$	$(N \times V_0) \rightarrow S$
		$((N \rightarrow S) \times V_1) \rightarrow (S \rightarrow S)$
monadic lambda-abstract operator	λ	$(V_0 \times S) \rightarrow (N \rightarrow S)$
parentheses	$(,)$	none

Note that some of the symbols are categorially ambiguous, as indicated in the category column. Note the presence of subcategories V_0 (individual variables) and V_1 (monadic predicate variables).

2. Non-Logical Vocabulary

proper nouns	N
first-order function signs (for each $k \geq 0$):	$N^k \rightarrow N$
first-order predicates (for each $k \geq 0$):	$N^k \rightarrow S$
second-order function signs (for each $k \geq 0$):	$(N \rightarrow S)^k \rightarrow (N \rightarrow S)$
second-order predicates (for each $k \geq 0$):	$(N \rightarrow S)^k \rightarrow S$

As with all formal languages, the non-logical vocabulary is theory-specific. To specify a *particular* language of this type, one must specify its non-logical vocabulary — what specifically are the proper nouns, function signs, and predicates.

3. Rules of Formation

a. Singular-terms

every individual variable/constant is a singular-term;
 every proper noun is a singular-term;
 if ϕ is a first-order function sign of degree n , and τ_1, \dots, τ_n are singular-terms,
 then $\phi(\tau_1, \dots, \tau_n)$ is a singular-term;
 if τ is a singular-term, then $\bigcirc\tau$ is a singular-term.
 if Φ is a formula, and v is an individual variable, then $\iota v\Phi$ is a singular-term;
 nothing else is a singular-term.

b. First-Order Predicates

'=' is a first-order predicate of degree 2 [written in infix notation];
 every predicate variable/constant of degree 1 is a first-order predicate of degree 1;
 if Φ is a formula, and v is an individual variable,
 then $[\lambda v\Phi]$ is a first-order predicate of degree 1;
 for each $n \geq 0$, every *non-logical* first-order predicate of degree n
 is a first-order predicate of degree n ;
 if F is a second-order function sign of degree n ,
 and P_1, \dots, P_n are monadic first-order predicates,
 then $F(P_1, \dots, P_n)$ is a monadic first-order predicate.
 nothing else is a first-order predicate.

c. Second-Order Predicates

for each $n \geq 0$, every *non-logical* second-order predicate of degree n
 is a second-order predicate of degree n ;
 '=' is a second-degree predicate of degree 2 [written in infix notation];
 nothing else is a second-order predicate.

d. First-Order Function Signs

\bigcirc is a function sign of degree 1.
 for each $n \geq 0$, every *non-logical* first-order function sign of degree n
 is a first-order function sign of degree n ;
 nothing else is a first-order function sign.

e. Second-Order Function Signs

for each $n \geq 0$, every *non-logical* second-order function sign of degree n
 is a second-order predicate of degree n ;
 nothing else is a second-order function sign.

f. "Atomic" Formulas

if P is a first-order predicate of degree n , and τ_1, \dots, τ_n are singular-terms,
 then $P[\tau_1, \dots, \tau_n]$ is an atomic formula;
 if τ_1 and τ_2 are singular-terms,
 then $[\tau_1 = \tau_2]$ is an atomic formula;
 if \mathbb{P} is a second-order predicate of degree n , and P_1, \dots, P_n are monadic first-order predicates,
 then $\mathbb{P}[P_1, \dots, P_n]$ is an atomic formula;
 nothing else is an atomic formula.

g. Formulas

- every atomic formula is a formula;
- if Φ is a formula, then so are: $\sim\Phi$, $\Box\Phi$, $\Diamond\Phi$, $\circ\Phi$;
- if Φ and Ψ are formulas, then so are: $(\Phi \rightarrow \Psi)$, $(\Phi \leftrightarrow \Psi)$, $(\Phi \& \Psi)$, $(\Phi \vee \Psi)$;
- if Φ is a formula, and v is an individual variable, then $\forall v\Phi$ and $\exists v\Phi$ are formulas;
- if Φ is a formula, and V is a predicate variable, then $\forall V\Phi$ and $\exists V\Phi$ are formulas;
- if Φ is a formula, τ is a singular-term, and v is an individual variable,
 - then $(\tau/v)\Phi$ is a formula;
- if Φ is a formula, \mathbf{P} is a monadic predicate, and V is a monadic predicate variable,
 - then $(\mathbf{P}/V)\Phi$ is a formula.
- nothing else is a formula.

5. Rules of Derivation

Having presented the syntax of ML+P, we now turn to its semantics, which we present by way of derivation rules. Taking the rules of Quantified Modal Logic with Actuality for granted, we propose the following additional rules. Note that many rules will have names that duplicate names from first-order modal logic. When an ambiguity occurs, which specific rule is being employed will usually be apparent from the context.

1. Notational Conventions

In the following, Φ is any formula, v is any individual variable, c is any individual constant, V is any monadic-predicate variable. A, B, C are monadic-predicate *constants*, and $\mathbf{P}, \mathbf{Q}, \mathbf{R}$, are *closed* monadic predicates.³ Where α is any expression, and β is any variable, $\Phi[\alpha/\beta]$ is the formula that results when α replaces every occurrence of β that is free in Φ . A constant counts as old precisely when it occurs in a line that is neither boxed nor cancelled; otherwise, it counts as *new*. As usual, any rule that does not have an explicit index is understood so that both input and output are stated relative to the same index, and any rule that employs ‘ \equiv ’ is a bi-directional rule.

2. Quantifier Rules

Universal-Out ($\forall\text{O}$)	Existential-In ($\exists\text{I}$)
$\frac{\forall V\Phi}{\Phi[A/V]}$	$\frac{\Phi[A/V]}{\exists V\Phi}$
A is any monadic-predicate constant.	
Universal-Derivation (UD)	Existential-Out ($\exists\text{O}$)
$\begin{array}{l} \text{SHOW: } \forall V\Phi \\ \text{ SHOW: } \Phi[N/V] \\ \\ \end{array}$	$\frac{\exists V\Phi}{\Phi[N/V]}$
N is any <i>new</i> monadic-predicate constant.	

³ As usual, an expression \mathcal{E} counts as *closed* precisely when no variable occurs free in \mathcal{E} .

Quantifier Negation (QN)	
$\frac{\sim \forall V \Phi}{\exists V \sim \Phi}$	$\frac{\sim \exists V \Phi}{\forall V \sim \Phi}$
Tilde-Universal-Out ($\sim \forall O$)	
$\frac{\sim \forall V \Phi}{\sim \Phi[N/v]}$	$\frac{\sim \exists V \Phi}{\sim \Phi[A/V]}$
<i>N</i> is any <i>new</i> monadic-predicate constant.	<i>A</i> is any monadic-predicate constant.

Notice that the second-order quantifier rules are just like their first-order counterparts. The difference, of course, is that we use upper-case letters (predicates) instead of lower-case letters (singular-terms). There is also an analogous demarcation of atomic predicates into those that are predicate constants, and those that are not. As with free first-order logic, we have the following principle.

We regard *constants* as purely intra-derivational devices. In particular, we count as constants only those atomic expressions that are introduced by way of UD or $\exists O$. In particular, no atomic predicate that occurs in the argument to be proven counts as a predicate constant.

3. Identity Rules

Reflexivity of = (R=) $\frac{}{P = P}$	Symmetry of = (S=) $\frac{P = Q}{Q = P}$	Transitivity of = (T=) $\frac{P = Q \quad Q = R}{P = R}$
Here, P, Q, R, are <i>closed</i> monadic predicates, atomic or molecular, constant or not.		

Leibniz's Law (LL)	
$\frac{\Phi[P/V] \quad P = Q}{\Phi[Q/V]}$	$\frac{\Phi[P/V] \quad Q = P}{\Phi[Q/V]}$
<i>V</i> must be modally free for both P and Q in Φ .	
α is modally free for β in Φ if and only if: β is a constant, or α does not occur inside the scope of a modal operator.	

Identity Repetition (=Rep)	Extensionality (Ext)
$\frac{A = B}{A = B} \quad /i$	$\frac{\forall x(\mathbf{P}x \leftrightarrow \mathbf{Q}x)}{\mathbf{P} = \mathbf{Q}}$
$\frac{A = B}{A = B} \quad /j \text{ (old)}$	
<i>A</i> and <i>B</i> are monadic-predicate constants .	P and Q are closed monadic predicates.

In other words, just as with identity among individual constants, identity among monadic-predicate constants is absolute.

4. Further Second-Order Rules

Scoped-Predicate Rule – Def (P/V)	Lambda-Conversion (λC)⁴
$\frac{(\mathbf{P}/V)\Phi}{\exists V\{V=\mathbf{P} \ \& \ \Phi\}}$	$\frac{[\lambda v\Phi][c]}{\Phi[c/v]}$
P is any closed monadic predicate.	c must be an individual constant .

Predicate Existence (E![P])	Predication Repetition (P-Rep)
$\frac{}{\exists X[X=\mathbf{P}]}$	$\frac{A[c]}{A[c]} \quad /i$
	$\frac{A[c]}{A[c]} \quad /j \text{ (old)}$
P is any closed monadic predicate.	<i>A</i> is a monadic-predicate constant ; <i>c</i> is an individual constant .

Note that the left rule is a zero-place rule – it has no input line. This rule says that, for every monadic predicate, there is a corresponding plurality.⁵ Note that the right rule is an absolute repetition rule, says the following, in effect.

if an individual is a member of a plurality, then it is so *absolutely*.

In other words, as with individuals, pluralities are absolute entities in second-order modal logic.

⁴ We consider a more general form of lambda-conversion in a later section.

⁵ Note: since we are using type-theoretic techniques, this sort of claim does not generate a Russell-like paradox.

6. How to Read Second-Order Quantifiers in ML+P

Once we have predicate variables, we can quantify over them, and produce formulas such as:

$$\exists X X[a]$$

How do we read such a formula? The following seem plausible.

a is something
 a does something

The first one seems misleading, because it is easy to confuse it with:

a is some thing

which is symbolized:

$$\exists x[a=x]$$

Even the ‘does’ reading may be misleading, since it suggests an *intensional* "object".

We might try to avoid the risk of unintentional intensionality by reading the formula as follows.

a is a member/element of some collective/plurality

But this sounds suspiciously like we are slipping into *set talk*.

If we want to avoid set talk in favor of pure *plural talk*, the following is a possibility.

there are **things** such that a is one of **those**

Here, we are taking the predicate variable ‘ X ’ to correspond to a plural demonstrative pronoun (e.g., ‘those things’, ‘those people’, ‘those numbers’, depending on the domain of discourse). This exactly parallels the fact that an individual variable corresponds to a singular demonstrative pronoun (e.g., ‘that thing’, ‘that person’, ‘that number’, depending on the domain of discourse).

7. Lambda-Conversion and Scoped-Terms

Recall that the lambda-conversion rule is restricted to constants. Based on this rule, we can construct the following derivation *schemata*, which shows how lambda-predicate application is related to scoped-terms. In particular, we have the following derived rule.

$\frac{[\lambda v \Phi][c]}{=}$ $(c/v)\Phi$ <p>c must be a constant</p>
--

Derivation Schemata:

(1)	$[\lambda v\Phi][c]$	Pr
(2)	$\text{SHOW: } (c/v)\Phi$	Def (τ/v)
(3)	$\text{SHOW: } \exists x\{x=c \ \& \ \Phi\}$	DD
(4)	$c=c$	R=
(5)	$\Phi[c/v]$	1, λC
(6)	$c=c \ \& \ \Phi[c/v]$	4,5,SL
(7)	$\exists x\{x=c \ \& \ \Phi\}$	6, $\exists I$
(1)	$(c/v)\Phi$	Pr
(2)	$\text{SHOW: } [\lambda v\Phi][c]$	DD
(3)	$\exists v\{v=c \ \& \ \Phi\}$	1,Def (τ/v)
(4)	$a=c \ \& \ \Phi[a/v]$	3, $\exists O$
(5)	$\Phi[c/v]$	4a,4b,LL
(6)	$[\lambda v\Phi][c]$	5, λC

Now, it is a matter of no particular importance whether we take the above rule as derived or fundamental. However, we might wish to take the above derived rule and significantly expand its authority, so that it becomes the official rule, but not restricted to constants. In that case, we have the following alternative, non-equivalent, formulation of lambda conversion.

Lambda Conversion – Expanded Version
$\frac{[\lambda v\Phi][\tau]}{\underline{\underline{(\tau/v)\Phi}}}$
τ is any closed singular-term.

In this way, we have a method of interpreting all lambda-predication sentences, not just the ones that involve constants. Notice that lambda-predication sentences are automatically *de re*.

8. Scoped-Predicates

One purpose in introducing second-order constructions into modal logic is to handle situations in which actuality occurs – but without actually having to use actuality. This is accomplished by *scoped-predicates*. Our official syntax goes as follows.

If P is a monadic predicate, and V is a monadic variable, then (P/V) is a scoped-predicate, A scoped-predicate, in turn, is a special kind of sentential adverb: if (P/V) is a scoped-predicate and Φ is a formula, then $(P/V)\Phi$ is a formula.

We read the formula

$$(P/V)\Phi$$

roughly as:

it is true of the things that are P that Φ

or:

the things that are P are such that Φ

or more colloquially:

the P's are such that Φ

Remember, given our restricted syntax, P must be monadic, although it is permitted to be complex.

The following is a simple example of how scoped-predicates can be read.

$(F/X) \forall y \{ Xy \rightarrow Gy \}$

the F's are such that: every **one of them** is G.

Casual perusal suggests that this is a rather oblique way to say:

every F is G

Let us see that this intuition is corroborated by our derivation system.

(1)	$(F/X) \forall y(Xy \rightarrow Gy)$	Pr
(2)	SHOW: $\forall x\{Fx \rightarrow Gx\}$	UCD
(3)	Fa	As
(4)	SHOW: Ga	DD
(5)	$\exists X \{ X=F \& \forall y(Xy \rightarrow Gy) \}$	1,Def (P/V)
(6)	$A=F \& \forall y(Ay \rightarrow Gy)$	5, \exists O
(7)	Aa	3,6a,LL
(8)	Ga	6b,7,QL
(1)	$\forall x(Fx \rightarrow Gx)$	Pr
(2)	SHOW: $(F/X)\forall y(Xy \rightarrow Gy)$	Def (P/V)
(3)	SHOW: $\exists X\{X=F \& \forall y(Xy \rightarrow Gy)\}$	DD
(4)	$\exists X[X=F]$	E![P]
(5)	$A=F$	4, \exists O
(6)	$\forall y(Fy \rightarrow Gy)$	1,AV
(7)	$\forall y(Ay \rightarrow Gy)$	5,6,LL
(8)	$A=F \& \forall y(Ay \rightarrow Gy)$	5,7,SL
(9)	$\exists X\{X=F \& \forall y(Xy \rightarrow Gy)\}$	8, \exists I [‘A’ is a constant]

9. Scoped-Predicates and Actuality

So far we have not produced a sentence that goes beyond ordinary logic in expressive power. In the present section, we break the bounds of ordinary logic. In particular, we show how scoped-predicates can be used to explicate formulas involving the sentential actuality operator. Consider the sentence

some F must be G

and consider the reading that involves actuality:

$\Box \exists x(\bigcirc Fx \& Gx)$

Remember ‘ \bigcirc ’ is the flexible actuality operator. Now consider the following plural re-formulation of the same sentence.

the F's are such that: one of them must be G

We can formalize this in second-order logic, using scoped-predicates, as follows:

$(F/X) \Box \exists y \{ Xy \& Gy \}$

which reads:

the F's are such that: necessarily, **one of them** is G

Here, the expression 'one of them' corresponds to ' $\exists y\{Xy \& \dots\}$ '

The following derivations show how the actuality-formula and the scoped-predicate-formula are equivalent.

(1)	$\Box\exists x(\Box Fx \& Gx)$	/0	Pr
(2)	SHOW: $(F/X)\Box\exists y(Xy \& Gy)$	/0	Def (P/V)
(3)	SHOW: $\exists X\{X=F \& \Box\exists y(Xy \& Gy)\}$	/0	DD
(4)	$\exists X[X=F]$	/0	E![P]
(5)	$A=F$	/0	4, $\exists O$
(6)	SHOW: $\Box\exists y(Ay \& Gy)$	/0	ND
(7)	SHOW: $\exists y(Ay \& Gy)$	/01	DD
(8)	$\exists x(\Box Fx \& Gx)$	/01	1, $\Box O$
(9)	$\Box Fa \& Ga$	/01	4, $\exists O$
(10)	Fa	/0	5a, $\Box O$
(11)	Aa	/0	5,10,LL
(12)	Aa	/01	P-Rep
(13)	$Aa \& Ga$	/01	9b,12,SL
(14)	$\exists y(Ay \& Gy)$	/01	13, $\exists I$
(1)	$(F/X)\Box\exists y(Xy \& Gy)$	/0	Pr
(2)	SHOW: $\Box\exists x(\Box Fx \& Gx)$	/0	ND
(3)	SHOW: $\exists x(\Box Fx \& Gx)$	/01	DD
(4)	$\exists X\{X=F \& \Box\exists y(Xy \& Gy)\}$	/0	1,Def (P/V)
(5)	$A=F \& \Box\exists y(Ay \& Gy)$	/0	4, $\exists O$
(6)	$\exists y(Ay \& Gy)$	/01	4b, $\Box O$
(7)	$Aa \& Ga$	/01	6, $\exists O$
(8)	Aa	/0	7a,P-Rep
(9)	Fa	/0	5a,8,LL
(10)	$\Box Fa$	/01	9, $\Box I$
(11)	$\Box Fa \& Ga$	/01	7b,10,SL
(12)	$\exists x(\Box Fx \& Gx)$	/01	11, $\exists I$

Next, we consider the dual problem of 'every-might', which occurs in the following sentence.

every F might be G

The reading that is interesting involves actuality, as follows.

$\Diamond\forall y(\Box Fy \rightarrow Gy)$

it is possible that everything that is actually F is G.

This can also be formulated using a scoped-predicate, as follows.

$(F/X)\Diamond\forall y(Xy \rightarrow Gy)$

the F's are such that: it might be that every **one of them** is G.

The following derivations show that these two formulas are equivalent in our second-order system.

(1)	$\diamond \forall y (\circ Fy \rightarrow Gy)$	/0	Pr
(2)	SHOW: $(F/X) \diamond \forall y (Xy \rightarrow Gy)$	/0	Def (P/V)
(3)	SHOW: $\exists X \{ X=F \ \& \ \diamond \forall y (Xy \rightarrow Gy) \}$	/0	DD
(4)	$\exists X [X=F]$	/0	E![P]
(5)	$A=F$	/0	4, $\exists O$
(6)	$\forall y (\circ Fy \rightarrow Gy)$	/01	1, $\diamond O$
(7)	SHOW: $\diamond \forall y (Ay \rightarrow Gy)$	/0	8, $\diamond I$
(8)	SHOW: $\forall y (Ay \rightarrow Gy)$	/01	UCD
(9)	Ab	/01	As
(10)	SHOW: Gb	/01	DD
(11)	Ab	/0	9, P-Rep
(12)	Fb	/0	5, 11, LL
(13)	$\circ Fb$	/01	12, $\circ I$
(14)	Gb	/01	6, 13, QL
(15)	$A=F \ \& \ \diamond \forall y (Ay \rightarrow Gy)$	/0	5, 7, SL
(16)	$\exists X \{ X=F \ \& \ \diamond \forall y (Xy \rightarrow Gy) \}$	/0	15, $\exists I$
(1)	$(F/X) \diamond \forall y (Xy \rightarrow Gy)$	/0	Pr
(2)	SHOW: $\diamond \forall y (\circ Fy \rightarrow Gy)$	/0	DD
(3)	$\exists X \{ X=F \ \& \ \diamond \forall y (Xy \rightarrow Gy) \}$	/0	1, Def (P/V)
(4)	$A=F \ \& \ \diamond \forall y (Ay \rightarrow Gy)$	/0	3, $\exists O$
(5)	$\forall y (Ay \rightarrow Gy)$	/01	4b, $\diamond O$
(6)	SHOW: $\forall y (\circ Fy \rightarrow Gy)$	/01	UCD
(7)	$\circ Fb$	/01	As
(8)	SHOW: Gb	/01	DD
(9)	Fb	/0	7, $\circ O$
(10)	Ab	/0	4, 9, LL
(11)	Ab	/01	10, P-Rep
(12)	Gb	/01	5, 11, QL
(13)	$\diamond \forall y (\circ Fy \rightarrow Gy)$	/0	12, $\diamond I$

Finally, we consider our fly-killing example.

Jay wants to kill a fly
 Jay wants it to be the case that Jay kills s fly

The trick here is that we do not want the following to be a logical consequence.

Jay wants it to be the case that there is a fly

Using the flexible actuality operator, we can symbolize it as follows.

$$\llbracket JW \rrbracket \exists x \{ \circ Fx \ \& \ KIx \}$$

Here, we note that the compound $\llbracket JW \rrbracket$ is a box-modality.

If we use plurals, via the technique of scoped-predicates, we can formulate it as follows.

$$(F/X) \llbracket JW \rrbracket \exists y \{ Xy \ \& \ KIx \}$$

Close scrutiny reveals that the relevant form is similar to ‘some F must be G’, which we have already examined. The key in the proposed translation is that ‘flies’ does not occur inside the modal operator $\llbracket JW \rrbracket$, so we cannot obtain the undesirable result that Jay wants there to be flies.

10. Why the Second-Order Quantifier Rules Require Constants

The rules of second-order quantification, like the rules of first-order modal logic, and like the rules of free (extensional) first-order logic, are officially restricted to constants.

In free logic, we do this as a simple way to effect the following stance. In particular, in free logic, we do not presume that a given singular-term denotes an element of the domain, but we do presume that variables and constants denote elements of the domain.

One might suspect that we have the same issue of denotation in second-order logic, but we must quickly discard this hypothesis in light of the predicate-existence rule E![P],

$$\frac{}{\exists X[X = \mathbb{P}]}$$

which claims, in effect, that every (monadic) predicate denotes!

So why do we still have the restriction on the quantifiers? The answer does not concern denotation; it concerns *modal scope*. If we concentrate on *non-modal* second-order logic, then the restriction is completely without teeth, because it can always be circumvented. But this restriction is critical in *modal* second-order logic, because it is used to block invalid arguments.

Consider dropping the restriction. Now, consider the following derivation, which is admissible by our relaxed rules.

(1)	$\Box \forall x(Fx \rightarrow Gx)$	Pr
(2)	$\text{SHOW: } (F/X)\Box \forall y(Xy \rightarrow Gy)$	Def (P/V)
(3)	$\text{SHOW: } \exists X\{X=F \ \& \ \Box \forall y(Xy \rightarrow Gy)\}$	DD
(4)	$F=F$	R=
(5)	$F=F \ \& \ \Box \forall x(Fx \rightarrow Gx)$	1,4,SL
(6)	$\exists X\{X=F \ \& \ \forall y(Xy \rightarrow Gy)\}$	5, \exists I

What is wrong with this derivation?

Well, the premise says:

it must be the case that: every F is G *de dicto*

The conclusion says:

the F's are such that: **they** must all be G *de re*

This is not a valid argument, going as it does from a *de dicto* necessity to a *de re* necessity. For example, the Constitution requires that all senators are citizens, but it requires of no particular senator that he/she is a citizen; no particular senator is mentioned in the Constitution. Similarly, it will always be the case that all senators are alive (presumably!), but no particular current senator will always be alive.

The problem that afflicts this argument is the same thing that afflicts the following similar argument involving singular-terms.

(1)	$\Box G[1xFx]$	Pr	
(2)	SHOW: $(1xFx/y)\Box Gy$	Def (τ/v)	
(3)	SHOW: $\exists y\{y=1xFx \ \& \ \Box Gy\}$	DD	
(4)	$1xFx = 1xFx$	R=	
(5)	$1xFx = 1xFx \ \& \ \Box G[1xFx]$	1,4,SL	
(6)	$\exists y\{y=1xFx \ \& \ \Box Gy\}$	5, \exists I	***

Now, we know that this inference has problems, even without the modal operator, because a true statement involving '1xFx' does not miraculously bring its denotation into existence; a statement involving the description '1xFx' can be true without '1xFx' having a denotation. So let us suppose an additional premise.

(0) $\exists x[x = 1xFx]$ Pr

But even presupposing, or even explicitly assuming, that '1xFx' is proper, we still have a modal problem.

For, the premise says:

it must be the case that: the F is G *de dicto*

While the conclusion says:

the F is such that: it must be G *de re*

This is an invalid inference, going as it does from a *de dicto* description to a *de re* description. How do we block this invalid inference? By making line (6) illegal!

And this is precisely how we block the earlier invalid inference – by prohibiting line (6)!