Part I

Models

Chapter 1
Predicate Models

1.1 Models

In TFL we use truth tables to calculate truth values of complex formulas based on the values of simpler ones, and see how the truth-value of formulas impact on that of others. In PL, because we are dealing with a more complex language, we need something more complex than truth tables: *models*. Models have a domain, predicate tables, relation tables, and identity tables. Models, like truth tables, are based on stipulations about the simplest bits of our language, namely names and predicates. They give possible ways information about predicates and names could be combined. With this basic information provided by models, we can use logic to find the truth value of more complex formulas. Like in TFL, we can also use models to figure out semantic notions that hold between formulas (consistency, validity, etc).

1.2 The Domain

The *domain* is the collection of things that we are talking about. So if we want to talk about people in Auckland, we define the domain to be people in Auckland. The quantifiers *range over* the domain. Given this domain, '$\forall x$' can be read roughly as 'Every person in Auckland is such that...' and '$\exists x$' as 'Some person in Auckland is such that...'.

In PL, the domain must always include at least one thing. We insist that each name must pick out exactly one thing in the domain. If we want to name people in places beside Auckland, then we need to include those people in the domain.

A domain must have *at least* one member. Each member of the domain must be named. A name must pick out *exactly* one member of the domain, but a member of the domain may be picked out by more than one name.

Because of the close relationship between names and objects in the domain, we use a notation that makes it clear which item of the domain names refer to. If a is a name in a symbolisation key, we will write $\texttt{a}$ for the object of the domain it refers to. Take for instance this symbolisation key:

a:	Andrew
e:	Emily
k:	Kim
Px:	x is a philosopher of science.
Lxy:	x works with y.
$x = y$:	x is identical to y.

Models for this key will have at most three items, because each item in the domain must be named, and the key only has three names. In a model, we would write a domain as:

$$\texttt{domain} = \{\texttt{a}, \texttt{e}, \texttt{k}\}$$

The name a, e and k respectively refer to items $\texttt{a}, \texttt{e}$ and $\texttt{k}$ of the domain. We could have a different model with a different domain in which two names refer to the same item:

$$\texttt{domain} = \{\texttt{a}, \texttt{e}(= \texttt{k})\}$$

This is a domain with two items $\texttt{a}$ and $\texttt{e}$, and the names e and k from the symbolisation key refer to the same item: $\texttt{e}$.

Logic Corner

When you start to do advanced work in logic, the assumption that every member of the domain must be named is dropped. If you think about it, that makes sense, because we don't have names for everything. If we did, every rock, every leaf of grass, indeed every subatomic particle would have a name. In this course, however, we will make our lives much simpler by requiring everything in the domain to have a name.

Mathematics Corner

We acknowledge mathematical reasons against the assumption that everything in a domain has a name. These go back to the work of Georg Cantor in the Nineteenth century about sizes of infinity. If a domain has an uncountable number of items (for instance the domain that contains all real numbers), then it is *impossible* to name them all! This follows from what some of you might know as the *pigeon-hole principle*. Such mathematical sophistication goes well beyond the practical purposes of this book.

Numbers and numerals provide an excellent way to appreciate the distinction between items and names. You may have wondered about the difference between the number 1 and the numeral 'one'. Consider the following domain and symbolisation key:

$$\texttt{domain} = \{1, 2, 3, 4\}$$

i:	One
ii:	Two
iii:	Three
iv:	Four

There are many ways we can assign the names 'i', 'ii', 'iii', and 'iv' to our domain. The Romans decided to give the item 1 the name 'i', 2 the name 'ii', 3 the name 'iii', and 4 the name 'iv'. In English we use the names 'one', 'two', 'three', and 'four'. Numerals (in any language) are merely names that refer to numbers. Numbers are the actual items in the mathematical domain.

1.3 Predicate tables

Carol has a small dog called Benji, and a Maine Coon cat called Louie. Maine Coon cats are big! We will analyse this case with this symbolisation key:

b:	Benji
l:	Louie
Bx:	x is big.
Dx:	x is a dog.
Mx:	x is a Maine Coon cat.

We record information about Carol's pets in a model using a *predicate table*:

`domain` $= \{$`b`$,$ `l`$\}$

	Bx	Dx	Mx
b	0	1	0
l	1	0	1

The model has a domain with two items, `b` (Benji) and `l` (Louie), and a predicate table that records information about them. Once we have a predicate table, we no longer need to work with the domain. We've collated the data in our table, so we can focus on how to treat the information it contains logically. We will no longer list the domain unless it is important to do so, although every model has a domain at its foundation. So let's focus on the predicate table:

	Bx	Dx	Mx
b	0	1	0
l	1	0	1

In the leftmost column, we list all the names of items in the domain, in this case the items `b` and `l`. The remaining columns, one for each predicate of the symbolisation key, tell us which items they apply to. For example, the second column tells us that b isn't a B ($B(b) = 0$), whereas l is a B ($B(l) = 1$). Hence, this predicate table models a case in which Benji is big, and Louie is big. We can use PL to read off all the basic information of this case:

$$\begin{array}{ccc} \neg Bb & Db & \neg Mb \\ Bl & \neg Dl & Ml \end{array}$$

We can use PL formulas to express the complex information contained in the table:

$$\begin{array}{cc} Db \wedge \neg Bb & Bl \wedge Ml \\ Bb \rightarrow \neg Db & Ml \vee \neg Bl \end{array}$$

Given a predicate table that record information about Benji and Louie, we can test the truth of complex statements, such as:

1. If Benji is a dog or a Maine Coon cat, then Benji is big.
2. Louie is not a big dog or a Maine Coon cat.

First we symbolise the statements in PL:

3. $(Db \vee Mb) \rightarrow Bb$
4. $\neg((Bl \wedge Dl) \vee Ml)$

Then we use the information from the predicate table to find the truth-values. We start by reading off the values from the predicate table:

$(Db$	$\vee$	$Mb)$	$\to$	Bb
1		0		0

After this step, we no longer need the predicate table. The rest is found by logic, using truth-tables. We start with the disjunction:

$(Db$	$\vee$	$Mb)$	$\to$	Bb
1	1	0		0

Finally, we find the value of the conditional:

$(Db$	$\vee$	$Mb)$	$\to$	Bb
1	1	0	0	0

Our logical analysis of Statement 1 tells us that it is false in the given model. We can similarly use PL to analyse Statement 2, resulting in this truth-table:

$\neg$	$((Bl$	$\wedge$	$Dl)$	$\vee$	$Ml)$
0	1	0	0	1	1

Our analysis of Statement 2 reveals that it is also false in this model.

We can now see the primary role of predicate tables in models: they lay out the basic information of the case being modelled, so that we can truth-functionally compose it into more complicated formulas. The predicate tables for a model in PL serve the same role as the valuation of the propositional variables in TFL.

This means that our counter-example will be represented by a model with predicate tables in PL rather than a simple valuation. Our methods for testing consistency, validity, etc will need to allow for this extra complexity.

Different cases will be modelled by different tables, which give us different information. Consider for instance this case:

	Bx	Dx	Mx
b	1	0	0
l	0	0	1

This predicate table models a case in which Benji, not Louie, is a big Maine Coon cat, and Louie is neither a dog nor a Maine Coon cat, and is not big. With this new predicate table, we again start by reading off the atomic information and plugging it in a truth table symbolising Statement 1:

$(Db$	$\vee$	$Mb)$	$\to$	Bb
0		0		1

Using logic to find the truth-value of the whole formula, we get:

$(Db$	$\vee$	$Mb)$	$\to$	Bb
0	0	0	1	1

Again, we use PL to analyse the symbolisation of Statement 2 for this case:

$\neg$	$((Bl$	$\wedge$	$Dl)$	$\vee$	$Ml)$
0	0	0	0	1	1

In this case, Statement 1 is true and statement 2 is false.

Predicate tables list all the possible ways information about predicates and names can be combined. Some are realistic, some aren't, but this is not something logic can assess. What logic can do is to calculate the information of complex formulas given the simple information contained in a predicate table. Logic even allows for weird cases that don't match up with reality, for instance:

	Bx	Dx	Mx
b	1	1	1
l	0	0	0

This is a predicate table that represents a case in which Benji is big, is a dog, and is a Maine Coon cat. It is not realistic, because dogs aren't cats, but logic is not concerned about that. What matters to logic is how the truth and falsity ascribed to simple facts combine into complex logical formulas.

1.4 Many-place Predicates

Predicate tables aren't too bad to understand when they only contain one-place predicates, but it gets messier when we consider two-place predicates. Consider a symbolisation key like:

Lxy: x loves y

Given what we said above, this symbolisation key should be read as saying:

- 'Lxy' and 'x loves y' are true of exactly the same things

So, in particular:

- 'Lxy' is to be true of x and y (in that order) iff x loves y.

It is important that we insist upon the order here, since love – famously – is not always reciprocated. The way we represent information in predicate tables will make the order explicit.

Suppose we are using the following names for people:

b:	de Beauvoir
l:	Lenin
m:	Marx
s:	Sartre

We can use a predicate table to represent who loves who in different cases, such as:

Lxy	b	l	m	s
b	1	0	0	0
l	1	1	1	1
m	0	0	1	0
s	0	0	0	0

Given this table, we read the information in a specific way. The top corner tells us what predicate we are considering, in this case the love predicate Lxy:

Lxy	b	l	m	s
b	1	0	0	0
l	1	1	1	1
m	0	0	1	0
s	0	0	0	0

A entry in the table tells us whether the name on the left column is related by the predicate to the name on the top row, in that order.

Lxy	b	l	m	s
b	1	0	0	0
l	1	1	1	1
m	0	0	1	0
s	1	0	0	0

The table tells us that Ll, m is true, because of the value 1 highlighted in green, which tells us that l (in the left column) is related to m (in the top row) by L. The table also tells us that Lm, l is false, because of the value 0 highlighted in magenta, which tells us that m is not related to l by L. Simply put, the predicate table models a case in which Lenin loves Marx, but Marx doesn't love Lenin. Further information can be read in the same way, for instance that Sartre loves de Beauvoir and no-one else, and that de Beauvoir loves herself and no-one else.

A model and its predicate tables can only tell us of one possible case amongst many others. Here's for instance a case in which everyone (in the domain) loves everyone:

Lxy	b	l	m	s
b	1	1	1	1
l	1	1	1	1
m	1	1	1	1
s	1	1	1	1

This simple table can tell many stories – that of mutual love, unfulfilled yearning, jealousy, betrayal, polyamory, and self-love. Combining it with predicates for sex, hate, and death, and you have the key elements for a heart-wrenching best-seller: 'PL: A Logical Romantic Tragedy'.

Logic Corner

Tables work well for one-place and two-place predicates. With three-place predicates, we would need three-dimensional tables, and for four-place predicates, we would need to step in the fourth-dimension. There are general ways to represent information for cases that are about for complex predicates, but you will only learn about those in upper stage logic.

Practice Exercises

Exercise A: Four superheroes walk into a bar...
This symbolisation key has four names:

b:	Batman
j:	Jessica Jones
s:	Spiderman
w:	Wonder Woman

Provide four models, with one, two, three, and four items in the domain.
Exercise B: Given this predicate table:

	Fx	Gx	Hx
a	1	0	1
b	0	1	0
c	1	1	1
d	0	0	0

What are the truth-values of the following formulas:

1. Fa, Gb, Gd.
2. $\neg Fb, \neg Gb, \neg Hb$.
3. $Fa \wedge Ga, Fa \vee Ga, Ga \wedge (Fa \wedge Ha)$.
4. $Fc \rightarrow Hc, Fc \rightarrow Gd, Gd \vee \neg Hb$.
5. $Ga \rightarrow (Hc \rightarrow Hd), Ha \rightarrow (Hc \rightarrow Hd)$.

Exercise C: Given this predicate table:

	Fx	Gx	Hx
a	1	1	0
b	0	1	1
c	1	0	0
d	0	0	1

What are the truth-values of the formulas from Exercise B?

Exercise D: For each group of formulas, provide a predicate table that makes them all true. If you cannot, explain why.

1. Fb, Gc, Gd, Pa.
2. $Fa, \neg Fc, \neg Gb, Gc, \neg He$.
3. $\neg Fa \wedge \neg Fb, Fa \vee Gc, Gc \rightarrow Fb$.
4. $Fa, Fa \rightarrow (\neg Ga \vee Gb), \neg(Gb \vee Fb)$.

5. $Ga \to (Hc \to Hd), Ha \to (Hc \to Hd)$.

Exercise E: For each group of formulas, provide a predicate table that makes them all false. If you cannot, explain why.

1. Fb, Gc, Gd, Pa.
2. $Fa, \neg Fc, \neg Gb, Gc, \neg He$.
3. $\neg Fa \wedge \neg Ga, Fa \vee Gc, Gc \to Fb$.
4. $Fa, Fa \to (\neg Ga \vee Gb), \neg(Gb \vee Fb)$.
5. $Ga \to (Hc \to Hd), Ha \to (Hc \to Hd)$.

Exercise F:
Given this two-place predicate table:

Lxy	b	l	m	s
b	1	0	0	0
l	1	1	1	1
m	0	0	1	0
s	0	0	0	0

What is the truth-value of the following formulas:

1. $Lb, b, Lb, s, Ls, s, Lm, l$.
2. $\neg Lb, b, \neg Lm, s, \neg Ll, m$.
3. $Ll, b \wedge Lm, m, Ls, s \vee Ls, l, Lm, s \wedge (Lm, s \vee Ll, l)$.
4. $Ll, m \to Lm, l, Lm, l \to Ll, m$.
5. $Ll, s \to (Lm, m \to (Ll, b \wedge Lm, s))$.

Exercise G: Given these predicate tables:

	Px	Qx
a	0	0
b	1	0
c	0	1
d	1	1

Rxy	a	b	c	d
a	1	0	0	1
b	0	1	1	1
c	0	0	1	0
d	0	1	0	0

Sxy	a	b	c	d
a	0	0	1	0
b	1	1	0	0
c	1	1	1	0
d	0	0	0	0

What is the truth-value of the following formulas:

1. $Pa, \neg Qc, Pa \vee \neg Qd$
2. $Pd \vee Ra, b, \neg Pa \wedge Sc, c, Sd, b \vee Ra, c$.
3. $Qb \to (Ra, c \wedge Sd, d), Pd \to \neg(Rc, c \to Sc, d)$.
4. $\neg((Ra, b \to Rc, a), Pd \wedge Qd, \neg Sc, b \to (Pc \wedge Qa)$.

Exercise H: For each group of formulas, provide a model that makes them all true. If you cannot, explain why.

1. $Pa, \neg Qc, Ra, b, \neg Sb, c$
2. $Pb \wedge Ra, b, \neg Pa \vee Ra, a, Sb, a \vee Rc, a.$
3. $Fa, Fa \rightarrow Gb, La, b \wedge \neg Gb.$
4. $\neg((Ra, b \rightarrow Rc, a), Pd \wedge Qd, \neg Sc, b \rightarrow (Pc \wedge Qa).$

Exercise I: For each group of formulas in Exercise H above, provide a model that makes them all false. If you cannot, explain why.

Chapter 2
Models with Identity

2.1 Identity tables

We talked of identity as a special predicate of PL. We write it a bit differently than other two-place predicates: '$x = y$' instead of 'Ixy' (for example). That it is a special predicate is most apparent by looking at *identity tables*. Identity tables are predicate tables that obey the specific principles that govern identity. These principles are:

Reflexivity	Everything is identical to itself.
Symmetry	If a is identical to b, then b is identical to a.
Transitivity	If a is identical to b, and b is identical to c, then a is identical to c.

Mathematics Corner

A relation that is reflexive, symmetric and transitive is called an *equivalence relation.*

Reflexivity That everything is identical to itself is represented in identity tables by (always) having a diagonal of 1's:

$=$	a	b	c
a	1		
b		1	
c			1

The values highlighted in green represent that $a = a$, $b = b$ and $c = c$. From now on, we will always include a diagonal of 1's in identity tables.

Symmetry That the order of names in an identity predicate does not matter is represented in identity tables by always copying the value for one ordering to the other. This has an interesting geometric effect on the identity table. You can imagine a mirror being placed over the main diagonal of the truth-table, with the values identical on both sides of the mirror.

$=$	a	b	c
a	1	1	0
b	1	1	
c	0		1

The values highlighted in green represent the symmetry between $a = b$ and $b = a$; both are true. The values highlighted in magenta represent the symmetry of $a = c$ and $c = a$; both are false.

Transitivity Transitivity isn't as easy to spot in identity tables as reflexivity or symmetry, but is just as important.

$=$	a	b	c
a	1	1	1
b		1	1
c			1

The values highlighted in green tell us that $a = b$, and that $b = c$. Transitivity of identity then tells us that $a = c$, which is highlighted in blue. Transitivity could manifest in other configurations of identity tables, such as:

$=$	a	b	c
a	1	1	1
b		1	
c		1	1

$$a = c$$
$$c = b$$
$$\therefore a = b$$

$=$	a	b	c
a	1		1
b	1	1	1
c			1

$$b = a$$
$$a = c$$
$$\therefore b = c$$

$=$	a	b	c
a	1	1	
b		1	
c	1	1	1

$$c = a$$
$$a = b$$
$$\therefore c = b$$

To keep track of transitivity, we recommend you write identity statements under the tables, as we just illustrated.

2.2 Identity & Predicate tables

There's a fundamental principle that describes how identity and predicate tables interact. It is attributed to Wilhelm Gottfried Leibniz in his *Discourse on Metaphysics*, and is called the *indiscernibility of identicals*:

$\boxed{\text{Identical things have exactly the same properties.}}$

For example, Spiderman can shoot webs, and Spiderman is identical to Peter Parker, so Peter Parker can also shoot webs. This principle tells us that the these tables are incompatible:

$=$	p	s
p	1	1
s	1	1

	Wx
p	0
s	1

They are incompatible because the identity table tells us that $p = s$, but the predicate table tells us that $Wp = 0$ and $Ws = 1$.

Identical implies Indiscernible

The principle of *indiscernibility of identicals* helps us to complete information described by partial predicate and identity tables. The principle of *indiscernibility of identicals* tells us that identical things have the same properties. This means these things have the same rows for all their predicate tables. Consider these partial tables:

$=$	a	b	c
a			1
b			
c			

	Px
a	1
b	
c	

We start by using reflexivity, symmetry, transitivity, and the information that $a = c$ to complete as much of the identity table as we can:

$=$	a	b	c
a	1		1
b		1	
c	1		1

	Px
a	1
b	
c	

Next, we apply the *indiscernibility of identicals* to complete the predicate table. We will make sure identical names have the same predicate entries.

Because $a = c$, the rows in the predicate tables for a and for c must be the same, we add a 1 for c under Px:

$=$	a	b	c
a	1		1
b		1	
c	1		1

	Px
a	1
b	
c	1

And this is all that we can do. As we don't have enough information for the remaining entries, we write '?' in the gaps:

$=$	a	b	c
a	1	?	1
b	?	1	?
c	1	?	1

	Px
a	1
b	?
c	1

Philosophy Corner

Leibniz also believed the reverse of the *indiscernibility of identicals*: if two things have the same properties, then they are identical. This is called the *identity of indiscernibles*. Our logic does not follow this principle; we allow that two *distinct* objects might have exactly the same properties.

WARNING We'd really like you to avoid a common mistake when completing tables. If you know that $a = b$ and Pa, then you also know that Pb. But you *can't* reason from $a \neq b$ and Pa to $\neg Pb$. For example John Key and Helen Clark are not the same person, but they are both past prime ministers of New Zealand. Suppose you are given those tables:

$=$	j	h
j		0
h		

	Px
j	1
h	

You can use reflexivity and symmetry to complete the identity table, but that's it. All you have is that $j \neq h$ and that Pj, which leaves open whether Ph or not:

$=$	j	h
j	1	0
h	0	1

	Px
j	1
h	?

Discernible implies Non-identical

When completing tables, you can also use negative information with the *converse indiscernibility of identicals* principle:

Things that have different properties are non-identical.

For example, Serena Williams has won 23 grand slams, and Roger Federer has won 20, so Serena Williams is not identical to Roger Federer. To apply the *converse indiscernibility of identicals* principle, we look for rows that have different entries. For example, consider these partial tables:

$=$	a	b	c
a			
b			
c			

	Px
a	1
b	
c	0

We start with the identity table, using reflexivity, which we can always apply:

$=$	a	b	c
a	1		
b		1	
c			1

	Px
a	1
b	
c	0

Next, we apply *converse indiscernibility of identicals.* We look for rows with different entries:

$=$	a	b	c
a	1		
b		1	
c			1

	Px
a	1
b	
c	0

Because Pa is true, but not Pc, we know that $a \neq c$, and we can include this information in the identity table:

$=$	a	b	c
a	1		0
b		1	
c	0		1

	Px
a	1
b	
c	0

And this is all that we can do. As we don't have enough information for any other entries, we put '?' in the remaining spaces:

$=$	a	b	c
a	1	?	0
b	?	1	?
c	0	?	1

	Px
a	1
b	?
c	0

WARNING We'd also like you to avoid a second common mistake when completing tables. If you know that Pa and $\neg Pb$, then you also know that $a \neq b$. But you can't reason from Pa and Pb to $a = b$. It's quite common for two different things to share a property. For example, John Key and Helen Clark are both past prime ministers of New Zealand, but they are not the same person! Suppose you are given these tables:

$=$	j	h
j		
h		

	Px
j	1
h	1

You can use reflexivity to enter values in the identity table, but that's it. You are given that Pj and Ph, but that doesn't inform you whether j is identical to h:

$=$	j	h
j	1	?
h	?	1

	Px
j	1
h	1

Now, before you accuse us of being conspiracy mongers claiming that John Key might be Helen Clark, let us assure you that's not what we are claiming. We are merely claiming that we haven't been provided enough information to tell them apart, yet.

So what would we need? Well, how do we ever tell people apart – we find something that's different or distinctive about them. For instance, Helen Clark is a member of the Labour party, and John Key is not. Let's add that information to our model:

$=$	j	h
j	1	
h		1

	Px	Lx
j	1	0
h	1	1

You will have noticed that we removed the '?' symbols. When we add any information, we always remove any '?' symbols, because the new information might help us to fill in the blanks. And in this case it does. By the *converse indiscernibiility of identical* principle, as $Lh = 1$ but $Lj = 0$, we know that $h \neq j$. With a little application of symmetry, we have:

$=$	j	h
j	1	0
h	0	1

	Px	Lx
j	1	0
h	1	1

And we've shown that Helen Clark and John Key were different (non-identical) prime ministers of New Zealand!

2.3 Identity & two-place predicate tables

Identity and two-place predicate tables interact in the same way as simple predicate tables. They are regulated by the *indiscernibility of identicals* principle and its converse. The *indiscernibility of identicals* principle tells us that identical things have the same properties and relations. For two-place predicate tables, this means that they have the same rows *and* columns. Consider this example, with the identity table already partially completed:

$=$	a	b	c
a			1
b		1	
c	1		1

Rxy	a	b	c
a	1		
b	0		
c		1	1

As we know that $a = c$, we ensure that a's and c's rows are identical:

$=$	a	b	c
a			1
b		1	
c	1		1

Rxy	a	b	c
a	1	1	1
b	0		
c	1	1	1

Next, we ensure that a's and c's columns are identical:

$=$	a	b	c
a			1
b		1	
c	1		1

Rxy	a	b	c
a	1	1	1
b	0		0
c	1	1	1

To apply the *converse indiscernibility of identicals* principle, we look for rows or columns that have different entries, as in:

$=$	a	b	c
a			
b			
c			

Rxy	a	b	c
a	1		
b	0		
c			

$a \neq b$

Sxy	a	b	c
a			
b	0		1
c			

$a \neq c$

With reflexivity and symmetry, we can complete our identity table:

$=$	a	b	c
a	1	0	0
b	0	1	?
c	0	?	1

Putting everything together, we can complete the following set of tables:

$=$	a	b	c
a		1	
b			
c			

	Fx	Gx
a		1
b	0	
c	0	

Rxy	a	b	c
a		0	1
b	0		
c		0	

We start with the identity table, completing as much as we can:

$=$	a	b	c
a	1	1	
b	1	1	
c			1

$a = b$

	Fx	Gx
a		1
b	0	
c	0	

Rxy	a	b	c
a		0	1
b	0		
c		0	

We gathered the information that $a = b$, which allows us to apply the *indiscernibility of identicals* principle. We ensure that a's and b's rows are the same:

$=$	a	b	c
a	1	1	
b	1	1	
c			1

$a = b$

	Fx	Gx
a	0	1
b	0	1
c	0	

Rxy	a	b	c
a	0	0	1
b	0	0	1
c		0	

Next we ensure that a's and b's columns are the same:

$=$	a	b	c
a	1	1	
b	1	1	
c			1

$a = b$

	Fx	Gx
a	0	1
b	0	1
c	0	

Rxy	a	b	c
a	0	0	1
b	0	0	1
c	0	0	

This is as much information as we can extract from the *indiscernibility of identicals* principle.

You might be tempted to reason that because both b and c have 0 for predicate F, they must be identical. This is one of the errors we just warned you about. It's particularly tempting in this example, as the b and c rows agree for several predicates, including the two-place predicate. However, B and C might just be similar; only the identity table can tell us that two items are identical.

Next we turn to the *converse* principle, looking for entries in tables that allow us to differentiate items:

$=$	a	b	c
a	1	1	
b	1	1	
c			1

	Fx	Gx
a	0	1
b	0	1
c	0	

Rxy	a	b	c
a	0	0	1
b	0	0	1
c	0	0	

$a = b$ $b \neq c$

We see that b and c do not have the same columns, which tells us that they are not equal. We add that information to the identity table, and apply the principle of symmetry:

$=$	a	b	c
a	1	1	
b	1	1	0
c		0	1

	Fx	Gx
a	0	1
b	0	1
c	0	

Rxy	a	b	c
a	0	0	1
b	0	0	1
c	0	0	

$a = b$ $b \neq c$

Now, we see from transitivity that $c \neq a$. Why? If it was true that $c = a$, we would get from transitivity that $c = b$, because $c = a$ and $a = b$ guarantees that $c = b$. But we already know that $c \neq b$. So it must be that $c \neq a$. (We also could have got this information by applying the *converse* principle to columns a and c, as we did for b and c.) This allows us to complete the identity table:

$=$	a	b	c
a	1	1	0
b	1	1	0
c	0	0	1

	Fx	Gx
a	0	1
b	0	1
c	0	

Rxy	a	b	c
a	0	0	1
b	0	0	1
c	0	0	

$a = b$ $b \neq c$, $a \neq c$

That's it! We get no further information, and we complete our tables by putting '?' in the remaining entries:

$=$	a	b	c
a	1	1	0
b	1	1	0
c	0	0	1

	Fx	Gx
a	0	1
b	0	1
c	0	?

Rxy	a	b	c
a	0	0	1
b	0	0	1
c	0	0	?

Practice Exercises

Exercise A: The following are partially-completed identity tables. For each identity table, fill in all the blank spaces that you can, using the principles of reflexivity, symmetry and transitivity. Put a '?' in spaces for which you don't have enough information.

1.

$=$	a	b
a		
b	0	1

2.

$=$	a	b
a	1	
b		

3.

$=$	a	b
a		
b	0	

4.

$=$	a	b
a		1
b		

5.

$=$	a	b	c
a		1	
b			
c	1		

6.

$=$	a	b	c
a	1		
b			
c		1	

7.

$=$	a	b	c
a			0
b			
c		0	

8.

$=$	a	b	c
a			0
b	1		
c			

Exercise B:

The following are partial identity and predicate tables. Fill in all the blank spaces in the tables using only the information already available. Put a '?' in entries for which you don't have enough information.

1.

$=$	a	b
a		
b		

	Fx	Gx
a		
b		

2.

$=$	a	b
a	1	0
b		

	Fx	Gx
a	1	0
b		1

3.

$=$	a	b	c
a			
b			
c			

	Fx	Gx	Hx
a			
b			
c			

4.

$=$	a	b	c
a			1
b			
c			

	Fx	Gx	Hx
a	1		0
b	0	1	
c		0	

5.

$=$	d	e	f
d			
e	1		
f			

	Ax	Bx	Cx
d		0	
e	0		1
f	1	0	

6.

$=$	f	j	n
f			
j			
n			

	Gx	Wx	Qx
f	1	0	0
j			1
n		1	1

7.

$=$	a	b	c
a		1	
b			1
c			

	Fx	Gx	Hx
a	1		
b		0	
c			1

Exercise C:

The following are partial identity and predicate tables. Fill in all the blank spaces in the tables using only the information already available. Put a '?' in entries for which you don't have enough information.

1.

$=$	a	b	c
a			
b			1
c			

	Fx	Gx
a		
b		0
c		

Rxy	a	b	c
a	0		
b			
c		1	

2.

$=$	a	b	c
a		1	
b			
c			

	Fx	Gx
a	0	
b		1
c		0

Rxy	a	b	c
a	0		1
b			
c	1		

3.

$=$	a	b	c
a			1
b			
c		1	

	Fx	Gx
a	0	
b		1
c		

Rxy	a	b	c
a	1		
b			
c			

4.

$=$	a	b	c
a			
b			
c			

	Fx	Gx
a	1	1
b	1	1
c	0	0

Rxy	a	b	c
a	1	1	0
b	0	1	0
c	0	1	1

5.

$=$	a	b	c
a			
b			
c		1	

	Fx	Gx
a		0
b	1	
c		0

Rxy	a	b	c
a	0	0	
b			
c			0

Sxy	a	b	c
a	1	0	
b			
c	0		1

6.

$=$	a	b	c
a			1
b			
c			

	Fx	Gx
a	0	
b	1	1
c		0

Rxy	a	b	c
a	1		
b		1	0
c		1	

Sxy	a	b	c
a	1		
b			0
c		1	

7.

$=$	a	b	c	d
a			1	
b				0
c		0		
d	1			

	Fx	Gx
a	0	
b	1	
c		
d		0

Lxy	a	b	c	d
a	0			
b		1	0	
c				
d		1		

Chapter 3

Models with Quantifiers

3.1 Quantified predicates

We finally come to the most intricate part of PL: quantifiers. What makes calculating the truth-value of quantifiers more difficult is that the result depends on multiple rows of a predicate table. A simple existentially quantified ($\exists$) formula is true if the unquantified formula is true on at least one row of the table, and a simple universally quantified ($\forall$) formula is true if the unquantified formula is true on all the rows. Let's start with an example:

x	Fx	Gx
a	0	1
b	1	1
c	0	1

This table represents a case in which b is F, but not a or c, whereas a, b, and c are all G. Because $Fb = 1$, we know that at least one item of the domain has the property F:

x	Fx	Gx
a	0	1
b	1	1
c	0	1

So $\exists x\, [Fx] = 1$. However, not all items in the domain have property F:

x	Fx	Gx
a	0	1
b	1	1
c	0	1

So we know $\forall x\, [Fx] = 0$.

What about G? The domain of the model has three items: a, b, and c. The predicate G is true of each of them:

x	Fx	Gx
a	0	1
b	1	1
c	0	1

Everything in the domain has the property G. Hence, $\forall x\,[Gx] = 1$.

Let's consider another model, in which everything is an F, but nothing is a G. In this model $\forall x\,[Fx] = 1$ and $\exists x\,[Gx] = 0$, just as we'd expect.

x	Fx	Gx
a	1	0
b	1	0
c	1	0

The truth-value of quantified formulas in models depends on everything in the domain. A simple universal formula $\forall x\,[Fx]$ is true in a model if everything is an F, and it is false if at least one thing is not an F:

$\forall x\,[Fx] = 1$

x	Fx
a	1
b	1
c	1

$\forall x\,[Fx] = 0$

x	Fx
a	1
b	0
c	1

A simple existential formula $\exists x\,[Fx]$ is true if at least one thing in the domain is an F, and it is false if everything is not an F:

$\exists x\,[F(x)] = 1$

x	Fx
a	0
b	1
c	0

$\exists x\,[F(x)] = 0$

x	Fx
a	0
b	0
c	0

But what about complex formulas like $\forall x\,[\neg Fx]$ or $\exists x\,[Fx \wedge Gx]$, or the even more complex $\forall x\,\exists y\,[\neg Lxy]$? We will need a more orderly process, which will be an extension of the complete truth tables that we used for TFL.

3.2 Single Quantifiers

Here is the first model we will use in this section:

x	Fx	Gx
a	0	1
b	1	1
c	0	1

To find the truth value of $\exists x\,[\neg Fx]$, we need to find the truth value of $\neg Fx$ for each item in the domain. We can use something like our familiar TFL truth-table. The initial columns are a little different, however. We have one column for each variable, and one row for each name it can take.

x	$\exists x$	$[\neg$	$Fx]$
a			
b			
c			

We start with the values for Fx for each item. From the model, we know $Fa = 0$, $Fb = 1$, and $Fc = 0$, which we enter in the truth-table:

x	$\exists x$	$[\neg$	$Fx]$
a			0
b			1
c			0

With these values recorded, we can calculate the values of $\neg Fx$ for each item, just as we would calculate negation in TFL:

x	$\exists x$	$[\neg$	$Fx]$
a		1	0
b		0	1
c		1	0

To find the value of the existential quantifier, we need to look for (at least) one row with a 1, which there is:

x	$\exists x$	$\neg$	Fx
a		1	0
b		0	1
c		1	0

We indicate that the existential quantifier is true like this:

	$\exists x$	$[\neg$	$Fx]$
a		1	0
b	$1\{$	0	1
c		1	0

The cell of the table highlighted in green is the item that makes the existential quantifier true, and the left brace '{' with the value 1 says that the formula is true of (at least) one of the bracketed items.

A similar table shows that $\forall x\, [\neg Fx] = 0$:

	$\forall x$	$[\neg$	$Fx]$
a		1	0
b	$0\{$	0	1
c		1	0

The cell highlighted in magenta indicates which item makes the quantified formula false. (The magenta and green cells are just for your reference – the large curly bracket and the truth value are all that you need to write).

Now that we've got the basics, we can look at some more interesting examples. We'll use a simpler model, but add a symbolisation key:

x	Cx
p	0
w	1

p:	Patrick
w:	Willi
Cx:	x is a cat.

These are the statements that we will be evaluating in our model:

1. Not everyone is a cat.
   $\neg\forall x\, [Cx]$
2. Willi is a cat, but not everyone is.
   $Cw \wedge \neg\forall x\, [Cx]$
3. Not everyone is a cat, but Willi is.
   $\neg\forall x\, [Cx] \wedge Cw$
4. Some creatures are cats and some aren't.
   $\exists x\, [Cx] \wedge \exists y\, [\neg Cy]$.

The PL truth table for Statement 1 looks like this:

x	$\neg$	$\forall x$	$[Cx]$
p	1	0 {	0
w			1

We can see that it's not just the quantifier column whose truth value is on a single row. The negation is also not dependent on which item the variable x is referring to, and so it does not need to be repeated on each row. This might be clearer in the PL truth table for Statement 2:

x	Cq	$\wedge$	$\neg$	$\forall x$	$[Cx]$
p	1	1	1	0 {	0
w					1

The rows in our PL truth tables represent the different items that each variable could be referring to, and so when a column is outside the scope of a quantified variable, it only needs to be calculated once across all those items. The braces '{' helps us to see that the information held across several rows is combined by the quantifier into a single row. However, sometimes there is a single row on both sides of a quantified sub-formula, as in Statement 3, and we need a new symbol to indicate the end of a quantifier, the ']':

x	$\neg$	$\forall x$	$[Cx]$	$\wedge$	Cq
p	1	0 {	0	] 1	1
w			1		

Finally, we will introduce a common mistake, and a useful logic technique for avoiding it. Statement 4 has two quantifiers, and we might think its PL table should look something like:

x	y	$\exists x$	Cx	$\wedge$	$\exists y$	$\neg$	Cy
p	p		0			1	0
p	w		0			0	1
w	p		1			1	0
w	w		1			0	1

However, we can't easily combine rows 1 and 3, nor rows 2 and 4, for our $\exists y$ quantifier. Instead, *because the two quantifiers are not nested*, we will rename the y variable to x, breaking our rule-of-thumb that every new quantifier gets a new variable.

x	$\exists x$	Cx	$\wedge$	$\exists x$	$\neg$	Cx
p	1 {	0	] 1	1 {	1	0
w		1			0	1

Now we have all the tools we need to consider multiple quantifiers.

3.3 Multiple Quantifiers

Let's start with a simple model with one predicate table:

Rxy	a	b
a	0	1
b	1	1

What is the value of $\forall x\,\forall y\,[Rxy \to \forall z\,[Ryz \to Rxz]]$ in this model?
The first step is to figure out how many rows we will need in our table. The difference with tables from the previous section is that we have quantifiers with overlapping scopes. This means that we must consider all possible combinations of items that the variables x, y, and z can take amongst a and b. There's a case in which all variables represent the item a, so we need to find the value of $Ra,a \to (Ra,a \to Ra,a)$; a case in which x represents a, y represents b and z represents a, so we need to find the value of $Ra,b \to (Rb,a \to Ra,a)$; and so on... In total, there are eight possible combinations, which means our empty PL table looks like this:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a								
a	a	b								
a	b	a								
a	b	b								
b	a	a								
b	a	b								
b	b	a								
b	b	b								

Now we can use our predicate table to record the values for the predicates, and then the connectives, in the innermost brackets:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a						0	1	0
a	a	b						1	1	1
a	b	a						1	0	0
a	b	b						1	1	1
b	a	a						0	1	1
b	a	b						1	1	1
b	b	a						1	1	1
b	b	b						1	1	1

Now, we need to find the truth-value for the innermost quantifier $\forall z$. For a universal quantifier to be true, as in the previous section, the formula must take the value 1 for each item in the domain *in which the value of the other variables is fixed.* This means that the universal quantifier will be true (for a set of rows where x and y are both fixed and z varies) if the value is 1 on all of those rows. Both x and y are fixed over the first two rows, so we group the first two rows of the table with a curly bracket, and put a line under it to demarcate it from the other cases. We repeat this for the remaining rows:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a					1	0	1	0
a	a	b						1	1	1
a	b	a					0	1	0	0
a	b	b						1	1	1
b	a	a					1	0	1	1
b	a	b						1	1	1
b	b	a					1	1	1	1
b	b	b						1	1	1

Next, we can add the values for the predicates, and then connectives, in the next layer of brackets:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a			0	1	1	0	1	0
a	a	b						1	1	1
a	b	a			1	0	0	1	0	0
a	b	b						1	1	1
b	a	a			1	1	1	0	1	1
b	a	b						1	1	1
b	b	a			1	1	1	1	1	1
b	b	b						1	1	1

For the quantifier $\forall y$, only the x remains constant, so we will group the first four rows of the table with a curly bracket, and the next four:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a		0	0	1	1	0	1	0
a	a	b						1	1	1
a	b	a			1	0	0	1	0	0
a	b	b						1	1	1
b	a	a		1	1	1	1	0	1	1
b	a	b						1	1	1
b	b	a			1	1	1	1	1	1
b	b	b						1	1	1

Finally, we find the value of $\forall x$, which needs to be true in all cases:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a	0	0	0	1	1	0	1	0
a	a	b						1	1	1
a	b	a			1	0	0	1	0	0
a	b	b						1	1	1
b	a	a		1	1	1	1	0	1	1
b	a	b						1	1	1
b	b	a			1	1	1	1	1	1
b	b	b						1	1	1

The formula $\forall x \forall y [Rxy \to \forall z [Ryz \to Rxz]]$ is therefore false in our model. The case in which x and z are set to a and y is set to b is a case in which we have $Ra, b = 1$ and $Rb, a = 1$, but $Ra, a = 0$, and this is enough to make the whole formula false in our model.

However, in a different model, it may be true. And we don't need to add or remove items from the domain. A small change to the entries in the predicate table should suffice:

Rxy	a	b
a	1	0
b	0	1

Go through the same process that we did above, with this new valuation. Once you've done that, check it against our result:

x	y	z	$\forall x$	$\forall y$	$[Rxy$	$\to$	$\forall z$	$[Ryz$	$\to$	$Rxz]]$
a	a	a	1	1	1	1	1	1	1	1
a	a	b						0	1	0
a	b	a			0	1	0	0	1	1
a	b	b						1	0	0
b	a	a		1	0	1	0	1	0	0
b	a	b						0	1	1
b	b	a			1	1	1	0	1	0
b	b	b						1	1	1

One important point, that we didn't emphasise when we set up the table, is that the order of the variables on the left hand side of the table must be the same order as the quantifiers; the variable from the first quantifier should be first, and so on. This ensures that the rows are grouped in the right order.

For our final example, we'll work with this model:

	Dx	Fx
a	0	0
b	1	1
c	1	1

Kxy	a	b	c
a	1	0	0
b	0	0	0
c	1	1	1

Is the formula $\forall x\,[Dx \to (\exists y\,[Kxy] \wedge Fx]$ true in our model? First, we need to figure out how many rows we will need in our table. As the formula has two variables that can each represent any of the three items in the model, we will need nine rows. We record the values for the innermost predicates. At this point we need to be careful, because only the predicate Kxy is in the scope of $\exists y$:

x	y	$\forall x$	$[Dx$	$\to$	$(\exists y$	$[Kxy]$	$\wedge$	$Fx]$
a	a					1		
a	b					0		
a	c					0		
b	a					0		
b	b					0		
b	c					0		
c	a					1		
c	b					1		
c	c					1		

We can now find the value of $\exists y$:

x	y	$\forall x$	$[Dx$	$\rightarrow$	$(\exists y$	$[Kxy]$	$\wedge$	$Fx]$
a	a					1		
a	b				1	0		
a	c					0		
b	a					0		
b	b				0	0		
b	c					0		
c	a					1		
c	b				1	1		
c	c					1		

The table tells us that for $x = a$, it is true that $\exists y\,[Kxy]$, and again for $x = c$, but for $x = b$ there is no y such that Kxy. Now we have exited the scope of the $\exists y$ quantifier, and so we will only be interested in the three distinct cases where $x = a$, $x = b$, and $x = c$. So the rest of the table will only use these three rows.
We can now complete the value for the conjunction:

x	y	$\forall x$	$[Dx$	$\rightarrow$	$(\exists y$	$[Kxy]$	$\wedge$	$Fx]$
a	a					1		
a	b				1	0	0	0
a	c					0		
b	a					0		
b	b				0	0	0	1
b	c					0		
c	a					1		
c	b				1	1	1	1
c	c					1		

The next step is the Dx predicate, and the conditional:

x	y	$\forall x$	$[Dx$	$\rightarrow$	$(\exists y$	$[Kxy]$	$\wedge$	$Fx]$
a	a					1		
a	b		0	1	1	0	0	0
a	c					0		
b	a					0		
b	b		1	0	0	0	0	1
b	c					0		
c	a					1		
c	b		1	1	1	1	1	1
c	c					1		

Finally, the truth-value of the universal quantifier $\forall x$, and the formula:

x	y	$\forall x$	$[Dx$	$\to$	$(\exists y$	$[Kxy]$	$\wedge$	$Fx]$
a	a					1		
a	b		0	1	1	0	0	0
a	c					0		
b	a					0		
b	b	0	1	0	0	0	0	1
b	c					0		
c	a					1		
c	b		1	1	1	1	1	1
c	c					1		

The formula $\forall x\,[Dx \to (\exists y\,[Kxy] \wedge Fx)]$ turns out to be false in our model. And the table even allows us to trace why. When $x = b$, the conditional is false because the conjunction is false, and that's because the formula $\exists y\,[Kxy]$ is false, and that's because there is no y such that Kby.

Practice Exercises

⋆ **Exercise A:** Consider the following model:

	Ax	Bx	Nx
a	0	1	1
b	1	1	0
c	0	1	0

Determine the truth value of the following formulas in this model:

1. Bc
2. $Ac \leftrightarrow \neg Nc$
3. $Nc \to (Ac \vee Bc)$
4. $\forall x\,[Ax]$
5. $\forall x\,[\neg Bx]$
6. $\exists x\,[Ax \wedge Bx]$
7. $\exists x\,[Ax \to Nx]$
8. $\forall x\,[Nx \vee \neg Nx]$
9. $\forall x\,[Nx \vee \neg Ax]$
10. $\exists x\,[Nx] \vee \neg \exists y\,[Ny]$
11. $\forall x\,[Ax] \vee \neg \forall y\,[Ay]$
12. $\forall x\,[Ax] \vee \forall y\,[\neg Ay]$
13. $\exists x\,[Nx] \vee \exists y \neg Ny$
14. $\exists x\,[Bx] \to \forall y\,[Ay]$
15. $\forall x\,[Ax \vee Nx] \to \neg \forall y\,[\neg By]$

Exercise B:

Given this model:

	Gx	Hx	Mx
c	0	0	1
e	0	1	0

Determine the truth value of the following formulas in this model:

1. Hc
2. He
3. $Mc \vee Me$
4. $Gc \vee \neg Gc$
5. $Mc \to Gc$
6. $\exists x\,[Hx]$
7. $\forall x\,[Hx]$

8. $\exists x\,[\neg Mx]$
9. $\exists x\,[Hx \wedge Gx]$
10. $\exists x\,[Mx \wedge Gx]$
11. $\forall x\,[Hx \vee Mx]$
12. $\exists x\,[Hx] \wedge \exists y\,[My]$
13. $\forall x\,[Hx \leftrightarrow \neg Mx]$
14. $\exists x\,[Gx] \wedge \exists y\,[\neg Gy]$
15. $\forall x\,\exists y\,[Gx \wedge Hy]$
16. $\exists x\,[\forall y\,[Gx \wedge Hy]]$
17. $\forall x\,\forall y\,[Gx \rightarrow \neg(Hy \wedge My)]$

Exercise C: Consider the following model:

Rxy	a	b	c
a	1	1	0
b	0	0	0
c	1	0	1

Sxy	a	b	c
a	0	1	0
b	1	1	1
c	0	1	0

Determine the truth value of the following formulas in this model:

1. $\exists x\,[Rxx]$
2. $\forall x\,[Rxx]$
3. $\exists x\,[Sxy]$
4. $\forall x\,[Sxy]$
5. $\exists x\,\forall y\,[Rxy]$
6. $\exists x\,\forall y\,[Rxy]$
7. $\forall x\,\exists y\,[Rxy]$
8. $\forall x\,\exists y\,[Sxy]$
9. $\exists x\,\forall y\,[Rxy]$
10. $\exists x\,\forall y\,[Ryx]$
11. $\forall x\,\forall y\,[(Rxy \wedge Ryx) \rightarrow Rxx]$
12. $\forall x\,\forall y\,[Rxy \rightarrow Sxy]$
13. $\exists x\,\forall y\,[Rxy \rightarrow Syx]$
14. $\exists x\,[Rxx \vee \neg Sxx]$
15. $\exists x\,[Rxx \vee \exists y\,[\neg Syy]]$
16. $\exists x\,\forall y\,[\neg Rxy]$
17. $\forall x\,[\exists y\,[Rxy] \rightarrow \exists z\,[Rxz]]$.
    NB This one requires a big table and we'll understand if you don't complete it. But how many rows do you need?

Exercise D:

Consider the following model:

Rxy	a	b
a	0	1
b	1	0

Sxy	a	b
a	1	0
b	0	1

Determine the truth value of the following formulas in this model:

1. $\exists x\,[Rxx]$
2. $\exists y\,[Syy]$
3. $\exists x\,[\neg Sxx] \vee \exists y\,[\neg Ryy]$
4. $\forall x\,[Rxx] \vee \forall y\,[Syy]$
5. $\forall x\,[\exists y\,[Rxy] \rightarrow \exists z\,[Rxz]]$.
6. $\forall x\,\forall y\,[Rxy \rightarrow \forall z\,[Szz]]$
7. $\exists x\,\exists y\,\exists z\,[Rxz \wedge (Sxy \vee Syx)]$
8. $\exists x\,\forall y\,[\exists z\,[Rxy \wedge (Ryx \rightarrow Rzz)]]$
9. $\exists x\,\exists y\,[Rxy \wedge \exists z\,[Sxx \vee Sxz]]$
10. $\forall x\,\forall y\,[Rxy \rightarrow (\exists z\,[Szx] \vee Syx)]$