

# Part I

## PL Truth Trees

---

## Chapter 1

# Reintroducing Truth Trees

---

In TFL a model was just an assignment of truth values to atomic formulas, so reasoning about all models was relatively straightforward: we could simply list them all in a complete truth table. In PL there isn't a simple method like complete truth tables for reasoning about consistency and validity. So we must use other methods, such as truth trees.

In this Part, we will extend our TFL truth tree method to allow for the extra complexity and symbols in PL. The tree method still tests whether a set of formulas is consistent, and we can still use it to test for properties like consistency and argument validity. If the root formulas are consistent, we will still generate a valuation from the truth tree; it will merely be a valuation for a model containing objects and predicates.

### Notation Corner

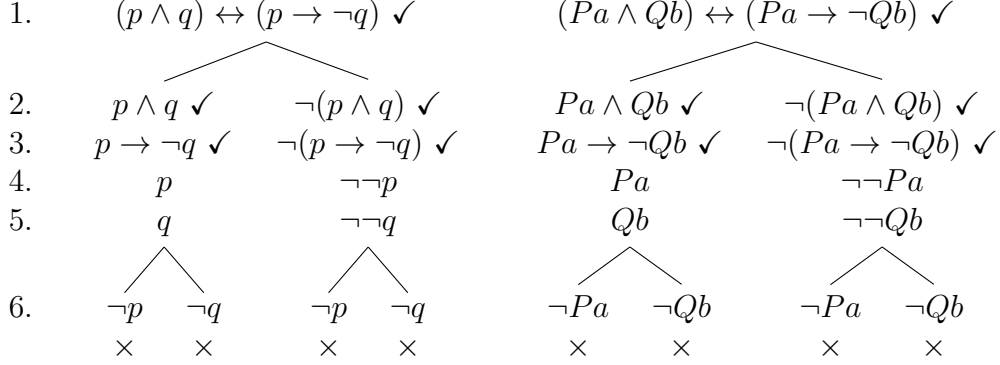
We have been representing predicates as  $P(a)$  or  $R(x, y)$ . But trees require us to write formulas repeatedly, and brackets can be confusing. It's fine to drop predicate brackets now and just write  $Pa$  or  $Rxy$ .

## 1.1 Unquantified Trees

PL without quantifiers behaves a lot like TFL. We have the same set of connectives, and atomic formulas such as  $Pa$  and  $Rbc$  behave exactly like atomic formulas such as  $p$  and  $q$  in TFL. It doesn't matter whether they are one-place predicates like  $Pa$  or many-place like  $Rbc$  or even  $Skdq$ . These atoms are true or false, and that's all the tree method cares about. So, if the root of our PL truth tree has no quantifiers, we can use exactly the same techniques as we did with TFL truth trees.

## Closed Trees

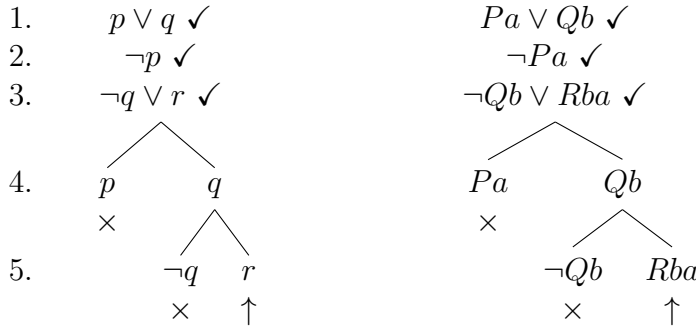
Compare the truth trees that test if the TFL formula  $(p \wedge q) \leftrightarrow (p \rightarrow \neg q)$  and the PL formula  $(Pa \wedge Qb) \leftrightarrow (Pa \rightarrow \neg Qb)$  are logical falsehoods:



These trees have parallel structures. The decomposition rules for conjunction, disjunction, conditional, biconditional, double negation, and the rules for creating a root, branching, and closing a branch, are identical.

## Open Trees

Similarly, compare the truth trees that test if the TFL formulas  $p \vee q$ ,  $\neg p$ ,  $\neg q \vee r$  and the PL formulas  $Pa \vee Qb$ ,  $\neg Pa$ ,  $\neg Qb \vee Rba$  are mutually consistent:



Again, the trees have parallel structures. When we read off the valuation which makes all the formulas true, we use the same process, but the valuations will look different.

The valuation for the TFL tree is  $p = 0, q = 1, r = 1$ .

The valuation for the PL tree is:

Domain =  $\{a, b\}$

	$P$	$Q$
a	0	?
b	?	1

$Rxy$	a	b
a	?	?
b	1	?

We can use tables to check that the valuations make all formulas true. In TFL:

$p$	$q$	$r$	$p \vee q$	$\neg p$	$\neg q \vee r$
0	1	1	1	0	0

In PL:

$Pa \vee Qb$	$\neg Pa$	$\neg Qb \vee Rba$
1	0	0

## 1.2 Quantified Trees

If we can remove all the quantifiers from a PL tree, we can simply use the existing TFL tree rules, as we have seen. This will be our goal. The problem is that we don't know which names (or objects) each variable should stand for in our valuations.

We will play it safe by assuming that an existential quantifier *might* be referring to a new object, and so we will always use a new name. (Remember that an object can have more than one name, so if the new name needs to apply to an object we've already named, that's fine). This means that our existential quantifier rule will add new information to our tree, unlike every other rule.

A universal quantifier will always be referring to all existing objects. But we don't know which objects exist, so we will confine ourselves to using old names; names that we know refer to existing objects. As a universal quantifier applies to all existing objects, it might be used several times. This means that our universal quantifier rule will be reusable, unlike every other rule.

We will also need to decompose negated quantifiers, in the same way that we decomposed negated conjunctions and so forth. Here, we will take a short-cut, to simplify our PL trees.

### Negated Quantifiers

Negated existential claims are quite similar to universals. They say that no instance is true, which is another way of saying that every instance is false. And in fact:

$$\neg \exists x \mathcal{A} \leftrightarrow \forall x \neg \mathcal{A}$$

Similarly, a negated universal is similar to an existential. It says that not all instances are true, so at least one instance is false:

$$\neg \forall x \mathcal{A} \leftrightarrow \exists x \neg \mathcal{A}$$

We will take advantage of these two equivalences to transform all negated quantified formulas into non-negated quantified formulas.

$\neg \exists x \mathcal{A}$		$\neg \forall x \mathcal{A}$
$\forall x \neg \mathcal{A}$	$i \neg \exists$	$\exists x \neg \mathcal{A} \quad i \neg \forall$

For example, if  $\neg \exists x [Fx]$  is in the tree, we can decompose it to  $\forall x [\neg Fx]$ . And likewise we can decompose  $\neg \forall z [Raz \rightarrow Pz]$  to  $\exists z \neg (Raz \rightarrow Pz)$ .

## Existential Quantifiers

Our tree rule for decomposing existential quantifiers replaces all instances of the existential variable in the formula with our new object name:

$\exists x \mathcal{A} \checkmark a$
$\mathcal{A} \langle \chi \Rightarrow a \rangle \quad i \exists a$
(where $a$ is new)

Just as  $\mathcal{A}$  stands in for any formula of PL, the  $\chi$  stands in for any variable, and  $a$  stands in for any name. The requirement that the name be *new* means that the name chosen for the substitution instance must be a name that has not appeared *anywhere* in the tree so far.

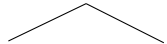
For example, consider this tree:

- |    |                                    |               |
|----|------------------------------------|---------------|
| 1. | $Fa$                               | Root          |
| 2. | $\exists x [\neg Fx] \checkmark b$ | Root          |
| 3. | $\neg Fb$                          | 2 $\exists b$ |

On line (2), we replace  $x$  with  $b$  (rather than  $a$ ), because  $a$  is a name that is already in use. We want to say that *something* is  $F$ , not necessarily that  $a$  is  $F$ . Of course,  $Fa$  could also be true: there could be more than one object that is  $F$ ; and it is also possible for  $a$  and  $b$  to be names for the same object ( $a = b$ ).

There is also a  $b$  next to the check mark in line (2). When we mark the existential quantifier as decomposed, we add the name that we've used to replace the quantified variable, so we can trace our work more easily.

Consider this more complex example:

1.	$\exists x [\exists y [Rxy] \rightarrow Rbx] \checkmark a$	Root
2.	$\exists y [Ray] \rightarrow Rba \checkmark$	1 $\exists a$
		
3.	$\neg \exists y [Ray] \checkmark \quad Rba$	2 $\rightarrow$
4.	$\forall y [\neg Ray]$	3 $\neg \exists$

On line (2), we decompose line (1) and can replace  $x$  with any letter except  $b$ , which is already used on line (1). On line (2), the main connective is not  $\exists$  but  $\rightarrow$ , so we can't decompose it using our existential tree rule. When we finally have a quantifier as the main connective on line (4), it's now a universal quantifier. We need another rule to decompose this formula.

## Universal Quantifiers

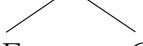
Our tree rule for decomposing universal quantifiers replaces all instances of the existential variable in the formula with an old object name:

$$\begin{array}{ll} \forall x \mathcal{A} \setminus a & \\ \mathcal{A} \langle \chi \Rightarrow a \rangle & i \forall a \\ & (\text{where } a \text{ is old}) \end{array}$$

For a name to be 'old' means that it has appeared *somewhere* in the tree.

Unlike all the other tree rules, we don't decompose a universally quantified formula and then check it off, never to use it again. Because more than one name can appear in a tree, each universal formula can be used more than once. We mark this by writing a  $\setminus$  instead of a  $\checkmark$  next to the universal formula.

Here is an example illustrating the importance of decomposing a universally quantified formula multiple times:

1.	$\forall x [Fx \wedge Gx] \setminus a, b$	Root
2.	$\neg Fa \vee \neg Gb \checkmark$	Root
3.	$Fa \wedge Ga \checkmark$	1 $\forall a$
4.	$Fa$	3 $\wedge$
5.	$Ga$	3 $\wedge$
		
6.	$\neg Fa \quad \neg Gb$	2 $\vee$
7.	$\times \quad Fb \wedge Gb \checkmark$	1 $\forall b$
8.	$Fb$	7 $\wedge$
9.	$Gb$	7 $\wedge$
	$\times$	

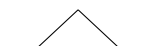
On line (3) we replace the universally quantified variable with  $a$  at line (1), and this helps to close one branch. But the other branch won't close until we take the  $b$  instance as well on line (7). Of course, we might have foreseen that we would need both a formula containing  $a$  and one containing  $b$  to close the tree, and replaced the quantified variable with  $b$  on line (4); the tree would have worked just as well. But in general, we want to delay acting on universal quantifiers for as long as possible, for reasons we'll explain shortly.

### 1.3 Completing your Tree



Although the connective, branching, and branch closing rules are identical to those in TFL, we will need to change our completion rule slightly.

The TFL tree completion rule, given in §??, said that a branch was complete if every complex formula has been decomposed, as indicated by a check mark. But universally quantified formulas may need to be decomposed several times – once for each name in the tree. We need our completion condition to ensure that we've taken *enough* instances.

For example, this should not count as a completed tree:

1.	$\forall x [Fx \rightarrow Ga] \setminus a$	Root
2.	$\neg Ga$	Root
3.	$Fb$	Root
4.	$Fa \rightarrow Ga \checkmark$	1 $\forall a$
		
5.	$\neg Fa \quad Ga$	4 $\rightarrow$
	$\times$	

The universal quantifier on line (1) can be decomposed using  $a$  and using  $b$ . If we did this, the tree would close:


1.	$\forall x [Fx \rightarrow Ga] \setminus a, b$	Root
2.	$\neg Ga$	Root
3.	$Fb$	Root
4.	$Fa \rightarrow Ga \checkmark$	1 $\forall a$
		
5.	$\neg Fa$	4 $\rightarrow$
6.	$Fb \rightarrow Ga \checkmark$	1 $\forall b$
		
7.	$\neg Fb$	6 $\rightarrow$
	$\times$	
	$Ga$	
	$\times$	

We don't want this tree to count as complete until it has decomposed line (1) using both  $a$  and  $b$ . A true universal formula is true of *every* object; we can't test that formula properly if we've only checked *some* objects.

For an open branch to be complete, every universally quantified formula in that branch must be decomposed using *every* name in the branch.

## 1.4 More PL Tree Examples

A tautology can never be false. A tree can test if a formula can be false. Is  $(\neg \exists x [Ax \wedge \neg Bx] \rightarrow \forall y [Ay \rightarrow By])$  a tautology?

1.	$\neg(\neg \exists x [Ax \wedge \neg Bx] \rightarrow \forall y [Ay \rightarrow By]) \checkmark$	Root
2.	$\neg \exists x [Ax \wedge \neg Bx] \checkmark$	1 $\neg \rightarrow$
3.	$\neg \forall y [Ay \rightarrow By] \checkmark$	1 $\neg \rightarrow$
4.	$\exists y \neg(Ay \rightarrow By) \checkmark a$	3 $\neg \forall$
5.	$\neg(Aa \rightarrow Ba) \checkmark$	4 $\exists a$
6.	$Aa$	5 $\neg \rightarrow$
7.	$\neg Ba$	5 $\neg \rightarrow$
8.	$\forall x \neg(Ax \wedge \neg Bx) \setminus a$	2 $\neg \exists$
9.	$\neg(Aa \wedge \neg Ba)$	8 $\forall a$
		
10.	$\neg Aa$	9 $\neg \wedge$
	$\times$	
	$\neg \neg Ba$	
	$\times$	



The tree shows that the negated formula can never be true. This means that the formula can never be false – it is a tautology.

Is the argument ‘No flasks are not made of glass so all flasks are made of glass’ valid?

$$\neg \exists x [Fx \wedge \neg Gx] \therefore \forall y [Fy \rightarrow Gy]$$

We check if the premises can be true while the conclusion is false:

1.	$\neg \exists x [Fx \wedge \neg Gx] \checkmark$	Premise
2.	$\neg \forall y [Fy \rightarrow Gy] \checkmark$	Neg Conc
3.	$\forall x [\neg (Fx \wedge \neg Gx)] \setminus a$	1 $\neg \exists$
4.	$\exists y [\neg (Fy \rightarrow Gy)] \checkmark a$	2 $\neg \forall$
5.	$\neg (Fa \rightarrow Ga) \checkmark$	4 $\exists a$
6.	$Fa$	5 $\neg \rightarrow$
7.	$\neg Ga$	5 $\neg \rightarrow$
8.	$\neg (Fa \wedge \neg Ga) \checkmark$	3 $\forall a$
9.	$\begin{array}{cc} & \wedge \\ \neg Fa & \neg \neg Ga \\ \times & \times \end{array}$	8 $\neg \wedge$

It is impossible for the premises to be true and the conclusion false, so there is no counter-example, and the argument is valid.

Proof that  $\forall y \exists x (Ryy \rightarrow Rxy)$  is a tautology:

1.	$\neg \forall y \exists x [Ryy \rightarrow Rxy] \checkmark$	Root
2.	$\exists y \neg \exists x [Ryy \rightarrow Rxy] \checkmark a$	1 $\neg \forall$
3.	$\neg \exists x [Raa \rightarrow Rxa] \checkmark$	2 $\forall a$
4.	$\forall x [\neg (Raa \rightarrow Rxa)] \checkmark a$	3 $\neg \exists$
5.	$\neg (Raa \rightarrow Raa) \checkmark$	4 $\exists a$
6.	$Raa$	5 $\neg \rightarrow$
7.	$\neg Raa$	5 $\neg \rightarrow$
	$\times$	

Proof that  $\forall x \forall y \forall z [(Rxy \wedge Ryz) \rightarrow Rxz], \forall x [\neg Rxx] \therefore \forall x \forall y [Rxy \rightarrow \neg Ryx]$  is valid:

1.	$\forall x \forall y \forall z [(Rxy \wedge Ryz) \rightarrow Rxz] \setminus a$	Premise
2.	$\forall x [\neg Rxx] \setminus a$	Premise
3.	$\neg \forall x \forall y [Rxy \rightarrow \neg Ryx] \checkmark$	Neg Conc
4.	$\exists x \neg \forall y [Rxy \rightarrow \neg Ryx] \checkmark a$	3 $\neg \forall$
5.	$\neg \forall y [Ray \rightarrow \neg Rya]$	4 $\exists a$
6.	$\exists y [\neg (Ray \rightarrow \neg Rya)] \checkmark b$	5 $\neg \forall$
7.	$\neg (Rab \rightarrow \neg Rba)$	6 $\exists b$
8.	$Rab$	7 $\neg \rightarrow$
9.	$\neg \neg Rba$	7 $\neg \rightarrow$
10.	$Rba$	9 $\neg \neg$
11.	$\forall y \forall z [(Ray \wedge Ryz) \rightarrow Raz] \setminus b$	1 $\forall a$
12.	$\forall z [(Rab \wedge Rbz) \rightarrow Raz] \setminus a$	11 $\forall b$
13.	$(Rab \wedge Rba) \rightarrow Raa \checkmark$	12 $\forall a$
14.	$\neg Raa$	2 $\forall a$
$\begin{array}{c} \swarrow \quad \searrow \\ \neg(Rab \wedge Rba) \checkmark \quad Raa \\ \quad \quad \quad \times \end{array}$		
15.	$\neg(Rab \wedge Rba) \checkmark$	13 $\rightarrow$
$\begin{array}{c} \swarrow \quad \searrow \\ \neg Rab \quad \neg Rba \\ \times \quad \quad \times \end{array}$		
16.	$\neg Rab$	15 $\neg \wedge$
	$\neg Rba$	
	$\times$	
	$\times$	

### Mathematics Corner

The above argument is a proof from mathematics. It says that any relation that is transitive and anti-reflexive (e.g., the greater-than ' $>$ ' and strict subset ' $\subset$ ' relations) must also be anti-symmetric.

The following argument is invalid:

$$\begin{aligned} & \forall x [\neg(\exists y [Sxy] \wedge \neg Ax)] \\ & \forall x [\exists y [Sxy] \rightarrow \neg Bx] \\ & \therefore \forall x [Sxx \rightarrow (Ax \wedge Bx)] \end{aligned}$$

1.	$\forall x [\neg(\exists y [Sxy] \wedge \neg Ax)] \setminus a$	Premise
2.	$\forall x [\exists y [Sxy] \rightarrow \neg Bx] \setminus a$	Premise
3.	$\neg \forall x [Sxx \rightarrow (Ax \wedge Bx)] \checkmark$	Neg Conc
4.	$\exists x [\neg(Sxx \rightarrow (Ax \wedge Bx))] \checkmark a$	3 $\neg \forall$
5.	$\neg(Saa \rightarrow (Aa \wedge Ba)) \checkmark$	4 $\exists a$
6.	$Saa$	5 $\neg \rightarrow$
7.	$\neg(Aa \wedge Ba)$	5 $\neg \rightarrow$
8.	$\neg(\exists y [Sya] \wedge \neg Aa) \checkmark$	1 $\forall a$
$\swarrow \quad \searrow$		
9.	$\neg \exists y [Sya] \checkmark$	8 $\neg \wedge$
10.	$\forall y [\neg Sya] \setminus a$	9 $\neg \exists$
11.	$\neg Saa$	10 $\forall a$
12.	$\times$	9 $\neg \neg$
$\swarrow \quad \searrow$		
13.	$\neg Aa$	7 $\neg \wedge$
14.	$\times$	2 $\forall a$
$\swarrow \quad \searrow$		
15.	$\neg \exists y [Say] \checkmark$	14 $\rightarrow$
16.	$\forall y [\neg Say] \setminus a$	15 $\neg \exists$
17.	$\neg Saa$	16 $\forall a$
	$\times$	

Fortunately the tree has produced a one-item counter-example. This will lead to a short counter-example and proof:

Domain = {a}	$\frac{Sxy}{a} \mid \frac{a}{1}$	$\frac{A \quad B}{a} \mid \frac{1 \quad 0}{0}$
--------------	----------------------------------	--

$\frac{x \quad y}{a \quad a} \mid \frac{\forall x [\neg(\exists y [Sxy] \wedge \neg Ax)]}{1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1}$	$\frac{x \quad y}{a \quad a} \mid \frac{\forall x [\exists y [Sxy] \rightarrow \neg Bx]}{1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0}$
---	--

$\frac{x \quad y}{a \quad a} \mid \frac{\neg \forall x [Sxx \rightarrow (Ax \wedge Bx)]}{1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0}$
--

$\forall x [\exists y [Rxy] \rightarrow Fx], \forall x [Fx \rightarrow \neg Rxx] \therefore \forall x \forall y [Rxx \vee \neg Rxy]$  is invalid:

1.	$\forall x [\exists y [Rxy] \rightarrow Fx] \setminus a, b$	Premise
2.	$\forall x [Fx \rightarrow \neg Rxx] \setminus a$	Premise
3.	$\neg \forall x \forall y [Rxx \vee \neg Rxy] \checkmark$	Neg Conc
4.	$\exists x [\neg (\forall y [Rxx \vee \neg Rxy])] \checkmark a$	3 $\neg \forall$
5.	$\neg (\forall y [Raa \vee \neg Ray]) \checkmark$	4 $\exists a$
6.	$\exists y [\neg (Raa \vee \neg Ray)] \checkmark b$	5 $\neg \forall$
7.	$\neg (Raa \vee \neg Rab) \checkmark$	6 $\exists b$
8.	$\neg Raa$	7 $\neg \vee$
9.	$\neg \neg Rab$	7 $\neg \vee$
10.	$Rab$	9 $\neg \neg$
11.	$\exists y [Ray] \rightarrow Fa \checkmark$	1 $\forall a$
<div style="text-align: center;">└──┬──</div>		
12.	$\neg \exists y [Ray] \checkmark$	11 $\rightarrow$
13.	$\forall y [\neg Ray] \setminus b$	12 $\neg \exists$
14.	$\neg Rab$	13 $\forall b$
15.	$\times \quad Fa \rightarrow \neg Raa \checkmark$	2 $\forall a$
<div style="text-align: center;">└──┬──</div>		
16.	$\neg Fa$	15 $\rightarrow$
17.	$\times \quad \exists y [Rby] \rightarrow Fa \checkmark$	1 $\forall b$
<div style="text-align: center;">└──┬──</div>		
18.	$\neg \exists y [Rby] \checkmark$	17 $\rightarrow$
19.	$\forall y [\neg Rby] \setminus a, b \quad \vdots$	18 $\neg \exists$
20.	$\neg Rba$	19 $\forall a$
21.	$\neg Rbb$	19 $\forall b$
	$\uparrow$	

While the branch under  $Fb$  might continue, it can't affect our open branch. Our counter-example and proof is:

Domain = $\{a, b\}$	$Rxy$	a	b	$F$
	a	0	1	
	b	0	0	

$x$	$y$	$\forall x [\exists y [Rxy] \rightarrow Fx]$	$\forall x [Fx \rightarrow \neg Rxx]$	$\neg \forall x \forall y [Rxx \vee \neg Rxy]$
a	a	$1 \left\{ \begin{array}{l} 1 \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right\} 1 \quad 1 \\ 0 \left\{ \begin{array}{l} 0 \\ 0 \end{array} \right\} 1 \quad ? \end{array} \right.$	$1 \left\{ \begin{array}{l} 1 \quad 1 \quad 1 \quad 0 \\ ? \quad 1 \quad 1 \quad 0 \end{array} \right.$	$1 \quad 0 \left\{ \begin{array}{l} 0 \left\{ \begin{array}{l} 0 \quad 1 \quad 1 \quad 0 \\ 0 \quad 0 \quad 0 \quad 1 \end{array} \right\} \\ 1 \left\{ \begin{array}{l} 0 \quad 1 \quad 1 \quad 0 \\ 0 \quad 1 \quad 1 \quad 0 \end{array} \right\} \end{array} \right.$
a	b			
b	a			
b	b			

---

## Chapter 2

# Tree Strategies

---

### 2.1 Order of Decomposition

You can always decompose the complex formulas of a truth tree in any order you want. The resulting trees will all eventually give the same (or equivalent) results. But some trees are shorter, or easier to find good strategies to complete, or easier to complete without making mistakes. Here's the advice we gave for TFL trees:

1. Non-branching rules before branching.
2. Close branches quickly.
3. Simple formulas before complex.
4. Biconditionals last.

All this still holds. However, we also have some quantifier-specific advice.

1. Negated Quantifiers are simple non-branching rules; do them early.
2. Existential Quantifiers after other non-branching rules.
3. Universal Quantifiers only when you have a clear plan for them.
4. Otherwise, delay Universal Quantifiers if there are many names.

This advice isn't perfect, but it will generally serve you well.

Existential quantifiers generate new information and new possibilities. Every existential quantifier will add a new name. Negated quantifiers will either become existential quantifiers, to be decomposed immediately, or universal quantifiers, to be planned for and used wisely (or often).

Because universal quantifiers can generate so many new formulas, you should either use them only when you have a clear plan, or when you are out of plans; at which point, use all the names, and see if anything useful appears!

Let's test if  $\forall x Rax, \forall x \forall y (Rxy \rightarrow Ryx) \therefore \forall y Rya$  is valid.

- |    |   |          |
|----|---|----------|
| 1. | $\forall x [Rax]$                           | Premise  |
| 2. | $\forall x \forall y [Rxy \rightarrow Ryx]$ | Premise  |
| 3. | $\neg \forall y [Rya]$                      | Neg Conc |

If we ignored (3) and decomposed our universals first, we'd get

- |                           |   |                  |
|---------------------------|---|------------------|
| 1.                        | $\forall x [Rax] \setminus a$                           | Premise          |
| 2.                        | $\forall x \forall y [Rxy \rightarrow Ryx] \setminus a$ | Premise          |
| 3.                        | $\neg \forall y [Rya] \checkmark$                       | Neg Conc         |
| 4.                        | $Raa$   | 1 $\forall a$    |
| 5.                        | $\forall y [Ray \rightarrow Rya] \setminus a$           | 2 $\forall a$    |
| 6.                        | $Raa \rightarrow Raa$                                   | 5 $\forall a$    |
| $\swarrow \quad \searrow$ |   |                  |
| 7.                        | $\neg Raa \quad Raa$                                    | 6 $\rightarrow$  |
| 8.                        | $\times \quad \exists y [\neg Rya] \checkmark b$        | 3 $\neg \forall$ |
| 9.                        | $\quad \neg Rba$  | 8 $\exists b$    |

The tree hasn't closed yet, and the new name  $b$  on line (8) means we need to rethink our use of lines (1), (2) and (5). If we instead delayed our universals until we had a plan, we might get:

- |                           |   |                  |
|---------------------------|---|------------------|
| 1.                        | $\forall x [Rax] \setminus b$                           | Premise          |
| 2.                        | $\forall x \forall y [Rxy \rightarrow Ryx] \setminus a$ | Premise          |
| 3.                        | $\neg \forall y [Rya] \checkmark$                       | Neg Conc         |
| 4.                        | $\exists y [\neg Rya] \checkmark b$                     | 3 $\neg \forall$ |
| 5.                        | $\neg Rba$  | 4 $\exists b$    |
| 6.                        | $Rab$   | 1 $\forall b$    |
| 7.                        | $\forall y [Ray \rightarrow Rya] \setminus b$           | 2 $\forall a$    |
| 8.                        | $Rab \rightarrow Rba$                                   | 7 $\forall b$    |
| $\swarrow \quad \searrow$ |   |                  |
| 9.                        | $\neg Rab \quad Rba$                                    | 8 $\rightarrow$  |
|                           | $\times \quad \times$                                   |                  |

With line (5) giving us  $\neg Rba$ , we could see that line (2), which switches the order of the names, could cause a contradiction with  $(Rab \rightarrow Rba)$  if we also had  $Rab$ . And line (1) could give us  $Rab$ . So we had a plan for which names to substitute into lines (1) and (2) – and with a little luck, our plan worked.

## 2.2 Truth Tree Complexities

### Strategically Selecting Names

In our last example, we had a plan for substituting our universal quantifiers by trying to create contradictions. This sort of strategy is important, particularly when you realise how long your tree could get if you just listed all possible substitutions. Lets try a more complex argument:

$$\forall x \exists y Rxy, \forall x \forall y Rxy \rightarrow Ryx, \forall x \forall y \forall z (Rxy \wedge Ryz) \rightarrow Rxz \therefore \forall x Rxx$$

We'll start by decomposing all the existentials as quickly as we can:

1.	$\forall x \exists y [Rxy] \setminus a$	Premise
2.	$\forall x \forall y [Rxy \rightarrow Ryx]$	Premise
3.	$\forall x \forall y \forall z [(Rxy \wedge Ryz) \rightarrow Rxz]$	Premise
4.	$\neg \forall x [Rxx] \checkmark$	Neg Conc
5.	$\exists x [\neg Rxx] \checkmark a$	4 $\neg \forall$
6.	$\neg Raa$	6 $\exists a$
7.	$\exists y [Ray] \checkmark b$	1 $\forall a$
8.	$Rab$	7 $\exists b$

Now we ask how we can get a contradiction with these formulas.  $\neg Raa$  is our only negative formula, so we need to get  $Raa$  from lines (2) or (3). If  $Raa$  was the consequent of line (2), the full formula would be  $Raa \rightarrow Raa$ , which doesn't seem useful? If  $Raa$  was the consequent of line (3), the full formula would be  $(Rab \wedge Rba) \rightarrow Raa$ . So we'd need  $Rab \wedge Rba$ . And we already have  $Rab$ ! How can we get  $Rba$ ? Well, line (2) can give us  $Rab \rightarrow Rba$ , and we already have  $Rab$ . We have a plan!!

9.	$\forall y [Ray \rightarrow Rya] \setminus b$	2 $\forall a$
10.	$Rab \rightarrow Rba$	9 $\forall b$
11.	$\neg Rab$ $Rba$	10 $\rightarrow$
12.	$\times$ $\forall y \forall z [(Ray \wedge Ryz) \rightarrow Raz] \checkmark b$	3 $\forall a$
13.	$\forall z [(Rab \wedge Rbz) \rightarrow Raz] \checkmark a$	12 $\forall b$
14.	$(Rab \wedge Rba) \rightarrow Raa$	13 $\forall a$
15.	$\neg(Rab \wedge Rba) \checkmark$ $Raa$	14 $\rightarrow$
16.	$\neg Rab$ $\neg Rba$	15 $\neg \wedge$
	$\times$ $\times$	

## Infinite Trees

Every decomposition of a TFL tree takes us closer to a completed tree, because we check of formulas as decomposed, and replace them with either one or two simpler formulas. TFL trees are guaranteed to be completed in a finite number of steps — they will either close, or every formula will be decomposed into an atomic or negated atomic formula.

However, the universal quantifier rule can be applied to the same formula once for every name in the tree, and so if we keep generating new names, it is possible for trees to continue indefinitely without closing.

Consider a tree with two formulas in its root:  $Raa, \forall x \exists y [Rxy]$ . We will have to decompose the universal, using the existing name ‘ $a$ ’. And that will give us  $\exists y [Ray]$ , which in turn will give us a new name ‘ $b$ ’, which we’ll use to decompose the universal quantifier...

1.	$Raa$	Root
2.	$\forall x \exists y [Rxy] \setminus a, b, c$	Root
3.	$\exists y [Ray] \checkmark b$	1 $\forall a$
4.	$Rab$	2 $\exists b$
5.	$\exists y [Rby] \checkmark c$	1 $\forall b$
6.	$Rbc$	4 $\exists c$
7.	$\exists y [Rcy] \checkmark d$	1 $\forall c$
8.	$Rcd$	5 $\exists d$
9.	$\vdots$	

As the pattern makes clear, this tree will never close, but it will also never be complete. It will just keep decomposing instances of an existential with new names, then decomposing those instances of the universal, which in turn require a new existential, and so on.

The tree will never close, and its open branch is describing a model where all the formulas are true, but it is an infinite model. There’s nothing illogical about that, but we certainly can’t use truth tables to check formulas in that model. The good news is that in PL whenever there is an infinite model that makes a finite list of formulas true, there is also a finite model that does the same. The bad news is that we can’t tell you how to find this finite model.

### Logic Corner

Congratulations, you’ve run into a limit of logic. PL is *semi-decidable*. We can always tell if a set of formulas is inconsistent, but we can’t always tell if a set of formulas is consistent – any algorithm will sometimes go into an infinite loop rather than stopping with an answer.



So, what should you do if your tree starts looping? First, check your tree decompositions, in case you've made a mistake. Then, once you are sure your tree is looping, stop extending the tree, draw a table with all the information you have *before the loop starts*, and manually, creatively, add more information to the table until you can make all the formulas true. We can't give you precise instructions on how to do this, because there is no algorithm that will always work.

However, usually (particularly with exercises in this course) you can produce a counter-example which is a model with one or two names. For example, with the tree above, a one-name model where  $Raa = 1$  makes both  $Raa$  and  $\forall x \exists y [Rxy]$  true. So would any two-name model with  $Raa = 1$  and  $Rba = 1$ , or  $Raa = 1$  and  $Rbb = 1$ . You can then check your counter-example using the techniques we discussed in Part ??.

Here is one counter-example, taken from the first 4 lines of the tree:

Domain = $\{a, b\}$			$x$	$y$	$Raa$	$\forall x$	$\exists y$	$[Rxy]$
	$Rxy$	$a$ $b$	$a$	$a$	1	1	1	1
			$a$	$b$				1
	$a$	1 1	$b$	$a$				1
	$b$	1 ?	$b$	$b$				?

### Logic Corner

In theory, to determine whether a set of formulas is consistent, or an argument valid, you don't need to produce a finite model. If a tree continues infinitely, there will always be an open branch. But in this course you must always produce a finite model.

## An Old New Name

There is one case where you can use a *new* name when decomposing a universal quantifier. Suppose that you have no existential quantifiers, and no names already mentioned. Then there are no old names, and you can't decompose any of your universal quantifiers. In this one case, you can (after double-checking), use a new name to decompose a universal quantifier. Why? Because we can assume that the model we are reasoning about is not empty. Empty models aren't interesting. Consider the seemingly valid argument that 'everything is made out of atoms, so something is made of atoms'. If there's nothing at all, then the premise could be true and the conclusion false. But truth trees assume there is always at least one object:

$$\forall x [Ax] \therefore \exists x [Ax]$$

1.	$\forall x [Ax]$	Premise
2.	$\neg \exists x [Ax] \setminus b$	Neg Conc
3.	$\forall x [\neg Ax] \checkmark b$	2 $\neg \exists$
4.	$\neg Ab$	3 $\forall b$
5.	$Ab$	1 $\forall b$
	$\times$	

Is  $\exists x \forall y [Ryy \rightarrow Rxy]$  a tautology?

1.	$\neg \exists x \forall y [Ryy \rightarrow Rxy] \checkmark$	Root
2.	$\forall x [\neg \forall y [Ryy \rightarrow Rxy]] \setminus a$	1 $\neg \exists$
3.	$\neg \forall y [Ryy \rightarrow Ray] \checkmark$	2 $\forall a$
4.	$\exists y [\neg Ryy \rightarrow Ray] \checkmark b$	3 $\neg \forall$
5.	$\neg (Rbb \rightarrow Rab) \checkmark$	4 $\exists b$
6.	$Rbb$	5 $\neg \rightarrow$
7.	$\neg Rab$	5 $\neg \rightarrow$
	$\uparrow$	

Is this valid: ‘All Unicorns have horns, so some Unicorns have horns’?

$$\forall x [Ux \rightarrow Hx] \therefore \neg \exists y Uy \wedge Hy$$

1.	$\forall x [Ux \rightarrow Hx] \setminus a$	Premise
2.	$\neg \exists y [Uy \wedge Hy] \checkmark$	Neg Conc
3.	$\forall y [\neg (Uy \wedge Hy)] \setminus a$	2 $\neg \exists$
4.	$Ua \rightarrow Ha \checkmark$	1 $\forall a$
5.	$\neg (Ua \wedge Ha) \checkmark$	3 $\forall a$
6.	$\begin{array}{cc} \neg Ua & Ha \end{array}$	4 $\rightarrow$
7.	$\begin{array}{cc} \neg Ua & \neg Ha & \neg Ua & \neg Ha \end{array}$	5 $\neg \wedge$
	$\uparrow$	$\times$

A counter-example is: Domain =  $\{a\}$ .  $\frac{\quad}{a} \mid \begin{array}{cc} Hx & Ux \\ ? & 0 \end{array}$

You can use new letters for your universals in other situations, but they will only make your tree longer, and never help it close or generate a counter-example. So don't. Just don't.

## 2.3 Common Student Errors

### Using Old Names

The most frequent errors for trees in PL involve ignoring the restrictions on names when taking instances. Particular formulas (existentials and negated universals) require a *new* name — if the name appears anywhere in the branch at the time the existential is taken (including below it), it is not suitable for substitution via these rules.

For example, this tree contains a mistake in a quantifier rule. See if you can spot it:

- |    |  |             |
|----|--|-------------|
| 1. | $\forall x \exists y [\neg Rxy] \setminus a$ |             |
| 2. | $Rab$  |             |
| 3. | $\exists y [\neg Ray] \checkmark b$          | 1 $\forall$ |
| 4. | $\neg Rab$                                   | 3 $\exists$ |
|    | $\times$                                     |             |

Line (3) here is fine; but at line (4), this tree uses the name ‘*b*’ to decompose the existential on line (3), even though ‘*b*’ had already appeared in line (2). The existential rule requires a new name. The correct version of this tree would remain open. (It will extend infinitely.)

### Main Connectives

You can only ever decompose the main connective of a formula. If you have a formula like  $\forall x \exists y (\neg Ryx) \wedge (Fa \rightarrow Fb)$  you can’t decompose the universal quantifier. This formula is a conjunction, so the only decomposition rule possible is that for conjunction.

It’s often tempting to decompose a quantifier that’s not the main connective. First, you can’t. The rules don’t work that way. Second, there are several ways that this can cause trouble. For instance, an existential quantifier within the scope of either a negation or the antecedent of a conditional will become a universal quantifier by the time the formula is decomposed. Second, if a quantifier is inside the scope of another quantifier, decomposing the inner quantifier first is effectively moving the inner quantifier outside the other quantifier. This quantifier scope shift is particularly problematic for an existential quantifier inside a universal quantifier.

It may help you to remember this by remembering our earlier example that ‘Every day someone is struck by lightning’ is true, but ‘Someone is struck by lightning every day’ is false, as different people are struck each day.

## Decomposing Universals Only Once

It's very easy to forget that a universal quantifier can be decomposed multiple times in a tree. And often a single use will close some or all branches. But if there's an open branch, you'll need to identify every name used in that branch, and make sure that every universal quantifier in that branch is decomposed for every name. This is important for populating the counter-example, and also because occasionally it will close a seemingly open branch.

## Negating Formulas in the Root

When testing consistency, we want to check if all the formulas can be true together. So no formula is negated. When testing validity, we want to check if all the premises can be true and the conclusion false (a counter-example). So we negate the conclusion only. When testing if a formula is a logical truth, we want to check if it can be false, so we negate the formula. It is very easy to tangle these up - particularly to forget to negate the conclusion of an argument. Then the information in the root of your tree is rotten. And a tree with a rotten root won't turn out well.

## Reading off Counter-examples incorrectly

When you read off a counter-example from an open branch, make sure that you only trace up the tree towards the root. Do not use any atomic formulas that are on other branches, even if they are also open. If a name doesn't appear on your chosen open branch, it's not needed in the counter-example. All the predicates in the root are needed, however.

Once you have your list of names from the open branch, and predicates from the root, draw up your tables – one for all the one-place predicates, and one for each two-place predicate. Then fill in the values for each atom in your open branch. Any cell in the tables that isn't filled with a 1 or 0 is then filled with a ?.

When you complete your partial predicate truth table, you should be able to fill in all the cells even with the missing truth values. If you need a truth value to calculate a row, check if you didn't decompose all names in one of the universal quantified formulas, as that's the most likely way to skip generating a needed atomic formula.

## Practice Exercises

★ **Exercise A:** Use a tree to test whether the following formulas are tautologies. If they are not tautologies, describe a model on which they are false.

1.  $\forall x \forall y [Gxy \rightarrow \exists z [Gxz]]$
2.  $\forall x [Fx] \vee \forall x [Fx \rightarrow Gx]$
3.  $\forall x [Fx \rightarrow (\neg Fx \rightarrow \forall y [Gy])]$
4.  $\exists x [Jx] \leftrightarrow \neg \forall x [\neg Jx]$
5.  $\exists x [Fx \vee \neg Fx]$
6.  $\forall x [Fx \vee Gx] \rightarrow (\forall y [Fy] \vee \exists x [Gx])$

★ **Exercise B:** Use a tree to test whether the following arguments are valid. If they are not, give a model as a counterexample.

1.  $Fa, Ga \therefore \forall x [Fx \rightarrow Gx]$
2.  $Fa, Ga \therefore \exists x [Fx \wedge Gx]$
3.  $\forall x \exists y [Lxy] \therefore \exists x \forall y [Lxy]$
4.  $\exists x [Fx \wedge Gx], Fb \leftrightarrow Fa, Fc \rightarrow Fa \therefore Fa$
5.  $\forall x \exists y [Gyx] \therefore \forall x \exists y [Gxy \vee Gyx]$

**Exercise C:** Translate each argument into PL, then use a tree to evaluate for validity. If they are invalid give a model as a counterexample.

1. Every logic student is studying. Patrick is not studying. Therefore, Patrick is not a logic student.
2. Kane Williamson is a New Zealand male athlete. Therefore, some athletes are New Zealand.
3. The All Blacks are going to win the game. Every team who wins the game will be celebrated. Therefore, the All Blacks will be celebrated.
4. The All Blacks are going to win the game. Therefore, the Springboks are not going to win the game.
5. All cats make Deborah sneeze, unless they are hairless. Some hairless cats are cuddly. Therefore, some cuddly things make Deborah sneeze.