

Symbolic Logic

J. Dmitri Gallow

Draft of January 5, 2016

Contents

1	Basic Concepts of Logic	1
2	Sentence Logic	5
2.1	Syntax for <i>SL</i>	6
2.1.1	Vocabulary	6
2.1.2	Grammar	6
2.1.3	Main Operators and Subformulae	7
2.2	Semantics for <i>SL</i>	10
2.2.1	The Meaning of the Statement Letters	10
2.2.2	The Meaning of ' \sim '	10
2.2.3	The Meaning of '&'	11
2.2.4	The Meaning of ' \vee '	11
2.2.5	The Meaning of ' \supset '	12
2.2.6	The Meaning of ' \equiv '	12
2.2.7	Determining the Truth-value of a wff of <i>SL</i>	12
2.3	Translation from <i>SL</i> to English	15
2.4	Translation from English to <i>SL</i>	18
2.4.1	Negation	18
2.4.2	Conjunction	19
2.4.3	Disjunction	20
2.4.4	The Material Conditional and Biconditional	21
2.5	What a Truth-Table Represents	21
2.6	<i>SL</i> Tautologies, Contradictions, and Contingencies	22
2.7	<i>SL</i> Equivalence	23
2.8	Disjunctive Normal Form and Expressive Completeness	24
2.8.1	Disjunctive Normal Form	24
2.8.2	Expressive Completeness	28
2.9	<i>SL</i> -Validity and <i>SL</i> -Invalidity	30
2.10	<i>SL</i> -Validity and <i>SL</i> -Tautologies	35
2.11	<i>SL</i> -Consistency & <i>SL</i> -Inconsistency	36
2.12	<i>SL</i> -Consistency and the Other Logical Properties of <i>SL</i>	38

3	Sentence Logic Trees	42
3.1	Truth Trees: The Basics	42
3.2	Rules for Truth Trees	45
3.3	Summary of Rules	53
3.4	Strategies for Applying Rules	54
3.5	<i>ST</i> -Consistency	55
3.6	Reading Truth-Value Assignments off of Open Branches	57
3.7	<i>ST</i> -Validity	61
3.8	<i>ST</i> -Tautologies, <i>ST</i> -Contradictions, and <i>ST</i> -Contingencies	64
3.8.1	<i>ST</i> -Tautologies	64
3.8.2	<i>ST</i> -Contradictions	65
3.8.3	<i>ST</i> -Contingencies	67
3.9	<i>ST</i> -Equivalence	68
4	Predicate Logic	70
4.1	The Language <i>PL</i>	70
4.1.1	The Syntax of <i>PL</i>	71
4.1.2	Semantics for <i>PL</i>	79
4.1.3	More on <i>PL</i> -Interpretations of Predicates	81
4.1.4	Truth on an Interpretation	83
4.1.5	Notation	86
4.2	<i>PL</i> -Validity	87
4.3	Proving <i>PL</i> -validity and the method of Semantic Proof	90
4.4	Translations from <i>PL</i> into English	92
4.4.1	Translating Atomic wffs of <i>PL</i>	92
4.4.2	Translating Simple Quantified wffs of <i>PL</i>	93
4.4.3	Translating More Complicated Quantified wffs of <i>PL</i>	94
4.5	Translations from English into <i>PL</i>	97
4.6	Overlapping Quantifiers and Relational Predicates	99
4.6.1	Changing the Order of the Quantifiers	100
4.6.2	Changing the Order of the Bound Variables	104
4.6.3	Translating ' $(\forall x)(\exists y)Rxy$ ' and ' $(\forall x)(\exists y)Ryx$ '	106
4.6.4	Changing the Order of the Quantifiers and the Order of the Bound Variables	107
5	Predicate Logic Trees	109
5.1	Notation and Terminology	109
5.2	Predicate Logic Trees	110
5.2.1	Rule for Universally Quantified Wffs	111
5.2.2	Rule for Existentially Quantified Wffs	115
5.2.3	Rules for Negations of Quantified Wffs	117
5.3	Strategies for Applying Rules	119
5.4	Sample Predicate Logic Trees	120
5.5	Logical Properties of <i>PL</i>	121
5.6	<i>PT</i> -Consistency	122
5.7	Reading Partial <i>PL</i> Interpretations off of Open Branches	124

5.8	<i>PT</i> Validity	125
5.9	<i>PT</i> Tautologies, <i>PT</i> Contradictions, and <i>PT</i> Contingencies	128
5.10	<i>PT</i> -Equivalence	131
5.11	Infinite Trees	133
6	Predicate Logic with Identity and Functions	140
6.1	The Language <i>PLI</i>	140
6.1.1	Preliminary Orientation	140
6.1.2	Syntax for <i>PLI</i>	141
6.1.3	Semantics for <i>PLI</i>	145
6.1.4	More on <i>PLI</i> -Interpretations of Function Symbols	147
6.1.5	Truth on a <i>PLI</i> Interpretation	150
6.2	Logical Notions of <i>PLI</i>	152
6.3	Trees for <i>PLI</i>	152
6.3.1	The Rule (=)	153
6.3.2	The Rule ($\neq \times$)	155
6.3.3	Completing Trees	156
6.4	<i>PTI</i> Consistency	160
6.5	<i>PTI</i> Validity	161
6.6	<i>PTI</i> Tautologies, Contingencies, and Contradictions	163
6.6.1	Properties of Identity	164
6.7	<i>PTI</i> Equivalence	166
6.8	Infinite <i>PLI</i> Trees	167
6.9	Number Claims	168
6.10	The Only	176
6.11	Definite Descriptions	177
7	Metatheory for Sentence Logic	179
7.1	Use & Mention	179
7.2	Object Language & Metalanguage	180
7.3	Metavariables	181
7.3.1	Use and Mention with Metavariables	182
7.4	Syntax and Semantics	183
7.5	Syntactic Validity and Semantic Validity	186
7.6	Soundness and Completeness of the Tree Method	187
7.7	Mathematical Induction	190
7.7.1	Terminology	191
7.7.2	Examples of Mathematical Induction	193
7.8	Varieties of Mathematical Induction	195
7.9	Choosing the Right Inductive Property	197
7.10	More Examples of Mathematical Induction	199
7.10.1	Number of Parentheses	201
7.10.2	Substitution of <i>SL</i> -equivalents	204
7.10.3	Duals	208

8	Soundness of the Tree Method for Sentence Logic	211
8.1	Preliminaries	211
8.2	The Proof in Broad Outline	213
8.3	Proof that the Tree Method for <i>SL</i> is Sound	215
9	Completeness of the Tree Method for Sentence Logic	225
9.1	The Proof in Broad Outline	226
9.2	Proof that the Tree Method for <i>SL</i> is Complete	227

Chapter 1

Basic Concepts of Logic

1. Logic is the study of *arguments*.
 - (a) An ARGUMENT is (for our purposes), just a collection of STATEMENTS (or DECLARATIVE SENTENCES), one of which is designated as the CONCLUSION, the remainder of which are designated as PREMISES.
 - (b) A statement (declarative sentence) is a sentence of which it makes sense to say that it is true or false.
 - i. It's true/false that it is Monday. ✓ \longrightarrow 'It is Monday' is a statement.
 - ii. It's true/false that close the door! \times \longrightarrow 'Close the door!' is not a statement.
2. There are many good-making features that an argument can have.
 - (a) The premises can give good (though not conclusive) reason to accept the conclusion.
 - (b) The premises can be true.
 - (c) The premises can be probable.
 - \vdots
3. In this course, we'll just be focusing on one good-making feature that an argument can have: the property of *deductive validity*.

DEDUCTIVE VALIDITY An argument is DEDUCTIVELY VALID if and only if ('iff') there is no possible scenario in which the premises are all true while the conclusion is false.

An argument is deductively invalid iff it is not deductively valid.

DEDUCTIVE INVALIDITY An argument is DEDUCTIVELY INVALID iff there is a possible scenario in which the premises are all true while the conclusion is false.

4. We may also characterize deductive validity in terms of the notion of a *counterexample*.

COUNTEREXAMPLE A COUNTEREXAMPLE to the deductive validity of an argument is a specification of a possibility in which the premises of the argument are all true yet the conclusion is simultaneously false.

Then,

DEDUCTIVE VALIDITY (2) An argument is DEDUCTIVELY VALID iff it has no counterexample.

DEDUCTIVE INVALIDITY (2) An argument is DEDUCTIVELY INVALID iff it has a counterexample.

5. What we want is a theory that tells us, of any given argument, whether it is deductively valid (or just 'valid') or deductively invalid.
6. Other important logical notions:

- (a) Logical properties of statements: Statements can be *logical tautologies*, *logical contradictions*, or *logical contingencies*.

Logical Tautology A statement $\lceil A \rceil$ is a LOGICAL TAUTOLOGY iff there is no possibility in which $\lceil A \rceil$ is false (*i.e.*, iff $\lceil A \rceil$ is true in every possibility).

LOGICAL CONTRADICTION A statement $\lceil A \rceil$ is a LOGICAL CONTRADICTION iff there is no possibility in which $\lceil A \rceil$ is true (*i.e.*, iff $\lceil A \rceil$ is false in every possibility).

LOGICAL CONTINGENCY A statement $\lceil A \rceil$ is a LOGICAL CONTINGENCY iff there is some possibility in which $\lceil A \rceil$ is true and some possibility in which $\lceil A \rceil$ is false.

- (b) A logical property of *pairs* of statements: pairs of statements can be *logically equivalent*.

LOGICAL EQUIVALENCE A pair of statements, $\lceil A \rceil$ and $\lceil B \rceil$, are LOGICALLY EQUIVALENT iff $\lceil A \rceil$ and $\lceil B \rceil$ are true in all the same possibilities and false in all the same possibilities (*i.e.*, iff it is impossible for $\lceil A \rceil$ and $\lceil B \rceil$ to have different truth-values).

- (c) Logical properties of *sets* of statements: A *set* of statements can be *logically consistent* or *logically inconsistent*.

Logical Property	Applies Only To
Deductive Validity	Arguments
Deductive Invalidity	Arguments
Logical Tautology	individual statements
Logical Contradiction	individual statements
Logical Contingency	individual statements
Logical Equivalence	pairs of statements
Logical Consistency	sets of statements
Logical Inconsistency	sets of statements

Figure 1.1: : Logical properties and the kinds of entities to which they apply.

LOGICAL CONSISTENCY A set of statements $\{\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner\}$ is LOGICALLY CONSISTENT iff there is some possibility in which $\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner$ are all true.

LOGICAL INCONSISTENCY A set of statements $\{\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner\}$ is LOGICALLY INCONSISTENT iff there is no possibility in which $\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner$ are all true.

7. Special cases of deductive validity.

- (a) If there's no possibility in which all of the premises are true (*i.e.*, if the premises are logically inconsistent), then there's no possibility in which all of the premises are true and the conclusion is false. So the following arguments are deductively valid.

1. John is taller than Mary.
 2. Mary is taller than John.
-
3. So, Dinosaurs are not extinct.

1. It is raining today.
 2. It is not raining today.
-
3. So, F.D.R. conquered Mesopotamia.

- (b) If there's no possibility in which the conclusion is false (*i.e.*, if the conclusion is a logical tautology), then there's no possibility in which the premises are all true and the conclusion is false. So the following arguments are deductively valid.

1. I hate jam.
2. Kesha is the voice of a generation.

3. So, if today's Monday, then today's Monday.

1. Hamsters eat only cheese.
2. Running is faster than flying.

3. So, either Obama's president or Obama's not president.

Chapter 2

Sentence Logic

The plan: we're going to construct an artificial language, call it '*SL*' (for 'sentence logic') within which we can be rigorous and precise about which arguments are valid and which are invalid. This, together with a method for translating from English into *SL* (and out of *SL* into English) will allow us to theorize about which English-language arguments are deductively valid and which are deductively invalid.

In general, we can specify a language by doing three things: 1) giving the vocabulary for the language, 2) giving the grammar of the language—that is, specifying which ways of sticking together the expressions from the vocabulary are grammatical, and 3) saying what the *meaning* of every grammatical expression is. For instance, in English, the vocabulary consists of all of the words of English. The grammar for English consists of rules saying when various strings of English words count as grammatical English sentences. 'Bubbie makes pickles' and 'Colorless green ideas sleep furiously' will count as grammatical sentences, whereas 'Up bouncy ball door John variously catapult' does not count as a grammatical sentence. Finally, the meaning of every English sentence is given by providing a dictionary entry for every word of English and providing rules for understanding the meaning of sentences in terms of the meanings of the words appearing in the sentence. The first two tasks are the tasks of specifying the *syntax* of the language. The final task is the task of specifying the *semantics* of the language.

$$\begin{array}{l} \text{SYNTAX} \left\{ \begin{array}{l} 1. \text{ Vocabulary} \\ 2. \text{ Grammar} \end{array} \right. \\ \text{SEMANTICS} \text{ —} 3. \text{ Meaning} \end{array}$$

That's exactly what we're going to do for our artificial language *SL*. However, our task will be much simpler than the task of specifying English, as we will have a far simpler

vocabulary, a far simpler grammar, and a far simpler semantics.

2.1 Syntax for *SL*

2.1.1 Vocabulary

The vocabulary of *SL* includes the following symbols:

1. All capital letters (known as *atomic sentences*, or *sentence letters*),

$$A, B, C, \dots, Y, Z$$

2. logical operators:

$$\sim, \&, \vee, \supset, \equiv$$

3. parentheses

$$(,)$$

Nothing else is included in the vocabulary of *SL*.

2.1.2 Grammar

However, only one—the third—is a *well-formed formula* (or ‘wff’) of *SL*. We specify what it is for a string of symbols from the vocabulary of *SL* to be a *well-formed formula* (or ‘wff’) of *SL* with the following rules.

- A) Any statement letter, by itself, is a wff—known as an *atomic wff*.
- ~) If ‘**P**’ is a wff, then ‘ $(\sim\mathbf{P})$ ’ is a wff—known as a *negation*; and ‘**P**’ is known as its *negand*.
- &) If ‘**P**’ and ‘**Q**’ are wffs, then ‘ $(\mathbf{P} \& \mathbf{Q})$ ’ is a wff—known as a *conjunction*; and ‘**P**’ and ‘**Q**’ are known as its *conjuncts*.
- ∨) If ‘**P**’ and ‘**Q**’ are wffs, then ‘ $(\mathbf{P} \vee \mathbf{Q})$ ’ is a wff—known as a *disjunction*; and ‘**P**’ and ‘**Q**’ are known as its *disjuncts*.
- ⊃) If ‘**P**’ and ‘**Q**’ are wffs, then ‘ $(\mathbf{P} \supset \mathbf{Q})$ ’ is a wff—known as a *conditional*; ‘**P**’ is known as its *antecedent*, and ‘**Q**’ its *consequent*.

- \equiv) If $\ulcorner \mathbf{P} \urcorner$ and $\ulcorner \mathbf{Q} \urcorner$ are wffs, then $\ulcorner (\mathbf{P} \equiv \mathbf{Q}) \urcorner$ is a wff—known as a *biconditional*; $\ulcorner \mathbf{P} \urcorner$ is known as its *left-hand-side*, and $\ulcorner \mathbf{Q} \urcorner$ its *right-hand-side*.
-) Nothing else is a wff.

Note: \mathbf{P} and \mathbf{Q} do not appear in the vocabulary of SL . They are not themselves wffs of SL . Rather, they are being used here as FORMULAE VARIABLES—they are variables whose potential values are formulae of SL .

All and only the strings of symbols that can be constructed by repeated application of the rules above are well-formed formulae. To show that $\ulcorner ((\sim(P \vee Q)) \supset R) \urcorner$ is a wff, we could walk through the following steps to build the formula up:

- a) $\ulcorner P \urcorner$ is a wff [from (A)]
- b) $\ulcorner Q \urcorner$ is a wff [from (A)]
- c) So, $\ulcorner (P \vee Q) \urcorner$ is a wff [from (a) and (b) and (\vee)]
- d) So, $\ulcorner (\sim(P \vee Q)) \urcorner$ is a wff [from (c) and (\sim)]
- e) $\ulcorner R \urcorner$ is a wff [from (A)]
- f) So, $\ulcorner ((\sim(P \vee Q)) \supset R) \urcorner$ is a wff [from (d), (e), and (\supset)]

The (\supset) rule requires that we include the outermost parentheses around the expression $\ulcorner (\sim(P \vee Q)) \supset R \urcorner$, and the (\sim) rule requires that we include parentheses around $\ulcorner \sim(P \vee Q) \urcorner$. However, I will adopt the standard convention of omitting the outermost parentheses and omitting the parentheses surrounding a negation, writing, e.g., $\ulcorner \sim(P \vee Q) \supset R \urcorner$ rather than $\ulcorner ((\sim(P \vee Q)) \supset R) \urcorner$. This convention is harmless, but you should bear in mind that, *strictly speaking*, formula like $\ulcorner \sim(P \vee Q) \supset R \urcorner$ are not wffs of SL .

2.1.3 Main Operators and Subformulae

Given the rules for wffs provided above, we can give a simple definition of what a wff's *main operator* is. The wff's *main operator* is just the operator associated with the last rule which would have to be applied if we were building the formula up by applying the rules for wffs above. For instance, if we want to know what the main operator is for the wff $\ulcorner (\sim P) \& Q \urcorner$, we would just imagine running through the following proof that $\ulcorner (\sim P) \& Q \urcorner$ is a wff of SL , by applying to the rules for well formed formulae, *i.e.*,

- a) $\ulcorner P \urcorner$ is a wff [from (A)]
- b) So, $\ulcorner (\sim P) \urcorner$ is a wff [from (a) and (\sim)]

- c) ' Q ' is a wff [from (A)]
 d) So, ' $((\sim P) \& Q)$ ' is a wff [from (b), (c), and (&)]

Here, the fact that we had to appeal to the rule (&) in the final step of building up ' $((\sim P) \& Q)$ ' tells us that & is the main operator. Imagine that we had tried to build up the formula in some other way. For instance, suppose we had attempted to *first* apply the rule (&) and *then* the rule (\sim). Then, our derivation would have gone like this.

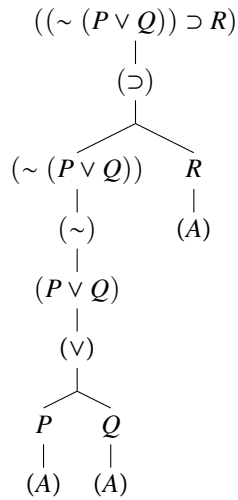
- a) ' P ' is a wff [from (A)]
 b) ' Q ' is a wff [from (A)]
 c) So, ' $(P \& Q)$ ' is a wff [from (a), (b), and (&)]
 d) So, ' $(\sim(P \& Q))$ ' is a wff [from (c) and (\sim)]

This is an entirely different wff. ' $(\sim(P \& Q))$ ' is not the same as ' $((\sim P) \& Q)$ '. While the main operator of ' $((\sim P) \& Q)$ ' is &, the main operator of ' $(\sim(P \& Q))$ ' is \sim .

We can also use the rules for wffs to give a definition of what a wff's *subformulae* are. p is a subformula of q if and only if, in the course of building up q by applying the rules for wffs, p appears on a line before q . So, for instance ' $(\sim P)$ ' is a subformula of ' $((\sim P) \& Q)$ ' (because it shows up on line (b) of that wff's derivation), whereas ' $\sim P$ ' is *not* a subformula of ' $(\sim(P \& Q))$ ' (since it does not show up at any point in that wff's derivation).

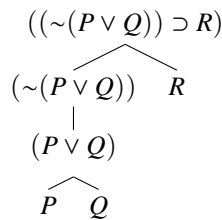
A formula's *immediate subformulae* are those wffs whose lines were appealed to in the final step of building to formula up. For instance, the immediate subformulae of ' $((\sim P) \& Q)$ ' are ' $(\sim P)$ ' and ' Q ', whereas the immediate subformula of ' $(\sim(P \& Q))$ ' is ' $(P \& Q)$ '. A wff's immediate subformulae are just those formulae on which the wff's main operator operates.

Another way of notating the proofs that certain formulae are wffs of SL is with SYNTAX TREES. For instance, we could represent our proof that ' $((\sim(P \vee Q)) \supset R)$ ' is a wff of SL with the following syntax tree.



This tree tells us, firstly, that ' P ' and ' Q ' are wffs of SL (by rule (A)). Then, by rule (\vee), ' $(P \vee Q)$ ' is a wff. Then, by rule (\sim), ' $(\sim (P \vee Q))$ ' is a wff. And, since ' R ' is a wff, by (A), ' $((\sim (P \vee Q)) \supset R)$ ' is a wff (by rule (\supset)).

If we want to leave out the rules, we can represent this syntax tree more simply as follows.



We can similarly write out the syntax trees for ' $((\sim P) \& Q)$ ' and ' $(\sim (P \& Q))$ ' like so.



These trees give us the syntactic structure of a wff of SL . They highlight what the parentheses were already telling us about what the main operator of the sentence is, what its subformulae are, and how the various subformulae are interrelated (how the sentence is built up out of its subformulae). For instance, the tree on the left tells us that the immediate subformulae of ' $((\sim P) \& Q)$ ' are ' $(\sim P)$ ' and ' Q '. And the tree on the right tells us that the immediate subformula of ' $(\sim (P \& Q))$ ' is ' $(P \& Q)$ '.

2.2 Semantics for *SL*

We now need to say something about the *meaning* of the wffs appearing in *SL*. Throughout, our assumption will be that what it is to understand the meaning of an expression is just to understand the circumstances in which it is true.

There are three components to the vocabulary of *SL*: the statement letters, the logical operators, and the parentheses. The parentheses do not add anything to the meaning of the sentences of *SL*. They merely serve as notational tools that help us avoid ambiguity. Put them aside. We must then say what the meanings of the statements letters are and what the meanings of the logical operators are.

2.2.1 The Meaning of the Statement Letters

Each statement letter represents a statement in English. The statement letter is true if and only if the statement in English is true. That is: statement letters inherit their meaning from their English translations.

2.2.2 The Meaning of ‘ \sim ’

If a wff ‘ \mathbf{P} ’, is true, then ‘ $\sim\mathbf{P}$ ’ is false. If a wff ‘ \mathbf{P} ’ is false, then ‘ $\sim\mathbf{P}$ ’ is true. To write this a bit more perspicaciously, we can use the letters ‘*T*’ and ‘*F*’ to stand for the truth-values true and false. Then, for any wff ‘ \mathbf{P} ’, if ‘ \mathbf{P} ’ is *T*, then ‘ $\sim\mathbf{P}$ ’ is *F*. If ‘ \mathbf{P} ’ is *F*, then ‘ $\sim\mathbf{P}$ ’ is *T*. We can summarize this with the following TRUTH TABLE.

P	$\sim\mathbf{P}$
<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>

This table tells us how the truth-value of a wff of the form ‘ $\sim\mathbf{P}$ ’ is determined by the truth-value of ‘ \mathbf{P} ’. If we understand the circumstances under which ‘ \mathbf{P} ’ is true, then the above definition gives us all that we need to understand the circumstances under which ‘ $\sim\mathbf{P}$ ’ is true. So we’ve said enough to say what the meaning of ‘ \sim ’ is.

Note that ‘ \mathbf{P} ’ is not a wff of *SL*—statement letters are not bolded. Rather, we are using the boldface ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ as variables ranging over the wffs of *SL*.

2.2.3 The Meaning of ‘&’

A conjunction is true iff both of its conjuncts are true. Using a truth-table, this means that:

P	Q	P & Q
<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>

This table tells us how the truth-value of a wff of the form ‘ $\mathbf{P \& Q}$ ’ is determined by the truth-values of ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’. If we understand the circumstances under which ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are true, then this definition gives us enough to understand the circumstances under which ‘ $\mathbf{P \& Q}$ ’ is true. So we’ve said enough to say what the meaning of ‘&’ is.

2.2.4 The Meaning of ‘ \vee ’

A disjunction is true iff at least one of its disjuncts is true.

P	Q	P \vee Q
<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>

This table tells us how the truth-value of a wff of the form ‘ $\mathbf{P \vee Q}$ ’ is determined by the truth-value of ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’. If we understand the circumstances under which ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are true, then this definition gives us enough to understand the circumstances under which ‘ $\mathbf{P \vee Q}$ ’ is true. So we’ve said enough to say what the meaning of ‘ \vee ’ is.

2.2.5 The Meaning of ‘ \supset ’

A conditional is true iff either its antecedent is false or its consequent is true.

P	Q	P \supset Q
<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>T</i>

As before, the above table gives us enough to understand the circumstances under which a wff of the form ‘ $\mathbf{P} \supset \mathbf{Q}$ ’ is true, assuming that we understand the circumstances under which ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are true. So this table defines the meaning of the operator ‘ \supset ’.

Note that this is the only binary operator which is not symmetric. That is, while ‘ $\mathbf{P} \& \mathbf{Q}$ ’ has the same meaning as ‘ $\mathbf{Q} \& \mathbf{P}$ ’, ‘ $\mathbf{P} \vee \mathbf{Q}$ ’ has the same meaning as ‘ $\mathbf{Q} \vee \mathbf{P}$ ’, and ‘ $\mathbf{P} \equiv \mathbf{Q}$ ’ has the same meaning as ‘ $\mathbf{Q} \equiv \mathbf{P}$ ’, ‘ $\mathbf{P} \supset \mathbf{Q}$ ’ *does not* have the same meaning as ‘ $\mathbf{Q} \supset \mathbf{P}$ ’.

2.2.6 The Meaning of ‘ \equiv ’

A biconditional is true iff its right hand side and its left hand side have the same truth-value.

P	Q	P \equiv Q
<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>

Again, this table gives us enough to understand the circumstances under which a wff of the form ‘ $\mathbf{P} \equiv \mathbf{Q}$ ’ is true, assuming that we understand the circumstances under which ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are true. So this table defines the meaning of the operator ‘ \equiv ’.

2.2.7 Determining the Truth-value of a wff of *SL*

If we know the truth-value of all the statement letters appearing in a wff of *SL*, then we can use our knowledge of the syntactic structure of the wff to determine its truth value. For instance, suppose that we know that ‘*P*’ is true and that ‘*Q*’ is false. Then, we know that ‘ $\sim P \& Q$ ’ is false, and that ‘ $\sim(P \& Q)$ ’ is true.



We can do the very same thing with truth-tables. For instance, to construct the truth-table for the wff ' $\sim P \& Q$ ', begin by writing out all the possible truth-values for P and Q .

P	Q	$\sim P$	$\& Q$
T	T		
T	F		
F	T		
F	F		

Then, copy the column of truth-values for P , placing it beneath every appearance of the statement letter P , and do the same for Q .

P	Q	$\sim P$	$\& Q$
T	T	T	T
T	F	T	F
F	T	F	T
F	F	F	F

Then, begin working your way up the syntactic structure of the sentence by calculating the truth-values of the subformulae appearing in the wff. We know how to calculate the truth-value of ' $\sim P$ ', given the truth-value of ' P ' (from the truth-table for \sim which tells us the meaning of ' \sim '), so do that first, placing the appropriate truth-values beneath the main connective of the subformulae ' $\sim P$ '.

P	Q	$\sim P$	$\& Q$
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	F

Now, we have to calculate the column of truth-values of ' $\sim P \& Q$ ', writing them out beneath the main connective of that wff—the ' $\&$ '. The truth-value of ' $\sim P \& Q$ ' is a function of the truth-values of ' $\sim P$ ' and ' Q ', and *not* the truth values of ' P ' and ' Q ', so we

must look at the bolded columns of truth-values below.

<i>P</i>	<i>Q</i>	\sim	<i>P</i>	$\&$	<i>Q</i>
<i>T</i>	<i>T</i>	F	<i>T</i>	T	T
<i>T</i>	<i>F</i>	F	<i>T</i>	F	F
<i>F</i>	<i>T</i>	T	<i>F</i>	T	T
<i>F</i>	<i>F</i>	T	<i>F</i>	F	F

Now, we can simply look to the truth-table for ‘ $\&$ ’ to figure out what column of truth-values ought to go beneath the ‘ $\&$ ’ in ‘ $\sim P \& Q$ ’. Since ‘ $\&$ ’ is the main operator of the wff, this tells us the column of truth-values associated with the wff ‘ $\sim P \& Q$ ’. To indicate that this column of truth-values is the column associated with the main operator of the wff ‘ $\sim P \& Q$ ’, we may put a box around this column.

<i>P</i>	<i>Q</i>	\sim	<i>P</i>	$\&$	<i>Q</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	F	<i>T</i>
<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	F	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	T	<i>T</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	F	<i>F</i>

This truth-table tells us how the truth-value of ‘ $\sim P \& Q$ ’ is determined by the truth-values of ‘*P*’ and ‘*Q*’. If ‘*P*’ is false and ‘*Q*’ is true, then ‘ $\sim P \& Q$ ’ is true. Otherwise, ‘ $\sim P \& Q$ ’ is false.

If we do the same thing with the wff ‘ $\sim(P \& Q)$ ’, we will arrive at the following truth-table.

<i>P</i>	<i>Q</i>	\sim	(<i>P</i>	$\&$	<i>Q</i>)
<i>T</i>	<i>T</i>	F	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	T	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	T	<i>F</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	T	<i>F</i>	<i>F</i>	<i>F</i>

This shows us how important it is to pay attention to the syntactic structure of the different wffs of *SL*—they end up making a difference to the *meaning* of those sentences. If we’re not careful with our parentheses, we’ll lose a big advantage of using a formal language—namely, that the sentences in *SL* are not ambiguous between different meanings.

2.3 Translation from *SL* to English

The meanings of \sim , $\&$, \vee , \supset , and \equiv are given by the truth-tables in the previous section. However, when we look at those meanings, it is difficult to not see some commonalities between these operators and some common English words. In particular, it appears that there's a very close connection between the meaning of ' \sim ' and the meaning of 'it is not the case that'; a very close connection between ' \vee ' and 'or'; a very close connection between ' $\&$ ' and 'and'.

Submitted for your approval: the following provides a translation guide from *SL* to English.

$\sim p$	\longrightarrow	It is not the case that p
$p \& q$	\longrightarrow	Both p and q
$p \vee q$	\longrightarrow	Either p or q
$p \supset q$	\longrightarrow	If p , then q
$p \equiv q$	\longrightarrow	p if and only if q

This translation guide requires some provisos. In the first place: there appears to be an important difference between the meaning of ' $p \supset q$ ' and 'if p , then q '. The difference is this: if ' p ' is false, then ' $p \supset q$ ' is automatically true, *no matter what statement q represents, and no matter what kind of connection there is between p and q* . However, we wouldn't ordinarily think that the sentence 'if John Adams was America's first president, then eating soap cures cancer' is true, just in virtue of the fact that 'John Adams was America's first president' is false. So it must be that 'if p , then q ' differs in meaning from ' $p \supset q$ '. I think that this is exactly right. However, there is still some close connection between the meanings of these two claims. To bring that connection out, suppose that I make the following claim:

If it's a weekday, then I'm on campus.

And suppose that Steve makes the claim,

If I'm on campus, then it's a weekday.

Think about the circumstances under which you could justly say that Steve or I had lied. If it's a weekday, but I'm not on campus, then I have lied. If, however, it's a weekday but Steve is not on campus, then he *hasn't* lied. After all, he never said that he would be on campus every weekday. He just said that, *if he's on campus*, then it's a weekday. But he did not commit himself to ever coming to campus at all. On the other hand, suppose that I'm on campus during the weekend. Then, you wouldn't be able to say that I had lied. For I never said that I would stay home during the weekend. I just said that, *if it's*

a *weekday*, then I'm on campus. However, if Steve is on campus during the weekend, then Steve *has* lied. After all, he said that he'd only be on campus on weekdays. Using '*D*' to represent the statement 'Dmitri is on campus', '*S*' to represent 'Steve is on campus' and '*W*' to represent 'it is a weekday', then it looks like the possibilities in which you can say that I have lied are just the possibilities in which the material conditional ' $W \supset D$ ' is false.

<i>D</i>	<i>W</i>	if <i>W</i> , then <i>D</i>	$W \supset D$
T	T	didn't lie	T
T	F	didn't lie	T
F	T	lied	F
F	F	didn't lie	T

And it looks like the possibilities in which you can say that Steve has lied are just the possibilities in which you can say that the material conditional ' $S \supset W$ ' is false.

<i>S</i>	<i>W</i>	if <i>S</i> , then <i>W</i>	$S \supset W$
T	T	didn't lie	T
T	F	lied	F
F	T	didn't lie	T
F	F	didn't lie	T

So, even though the translation isn't perfect, it's still pretty good. Moreover, even if a *SL* wff of the form ' $p \supset q$ ' might be better translated into English with 'Either it is not the case that *p* or *q*', it appears as though ' $p \supset q$ ' is the best possible *SL*-translation of the English 'if *p*, then *q*'. So that's how we'll be translating it here. But if you think the translation is less than perfect, you're absolutely correct. There are more advanced logics which attempt to give a better translation of the English conditional, but they are beyond the purview of this course.

In the second place: 'or' is used in English in two different senses. In one sense, called the 'inclusive or', a statement of the form '*p* or *q*' is true if and only at least one of '*p*' and '*q*' are true—that is, it is true if and only if either '*p*' is true, or '*q*' is true, or both are true. For instance, if I say to you 'either the elevator or the escalator is working', then I haven't lied to you if they are *both* working. To see this more clearly, think about the sentence 'if either the elevator or the escalator is working, then you will be in compliance with the Americans with Disabilities Act'. If both are working and you are not in compliance with the ADA, then I have lied to you. However, if 'either the elevator or the escalator is working' were false when they are both working, then I couldn't have lied to you.

Inclusive 'or': In the inclusive sense '*p* or *q*' means 'Either '*p*' or '*q*' or both.'

In another sense, called the ‘exclusive or’, a statement of the form ‘ p or q ’ is true if and only if at least and at most one of ‘ p ’ and ‘ q ’ are true. That is, in the exclusive sense, ‘ p or q ’ means ‘ p or q , but not both’. For instance, if your parent tells you, ‘Either you clean your room, or you’re grounded’, you clean your room, and your parent grounds you, then you can fairly complain that they lied.

Exclusive ‘or’: In the exclusive sense ‘ p or q ’ means ‘Either ‘ p ’ or ‘ q ’, but not both.’

When I say that ‘ $p \vee q$ ’ may be translated as ‘ p or q ’, I am using ‘or’ in its *inclusive* sense—that is, I am using it to mean ‘ p or q or both’.

‘ \vee ’ translates to the *inclusive* ‘or’

Let’s call the phrases on the right-hand-side of the translation guide above the *canonical logical expressions* of English. If the logical structure of an English statement is written in this form, then that statement is in *canonical logical form*. For instance, the following claim is in canonical logical form:

If both John loves Andrew and it is not the case that Andrew loves John,
then it is not the case that John and Andrew will be friends.

Because the sentence is in canonical logical form, it is simple to translate it into *SL*. We simply introduce the statement letters ‘ J ’, ‘ A ’, and ‘ F ’, where J = ‘John loves Andrew’, A = ‘Andrew loves John’, and F = ‘John and Andrew will be friends’. Then, the translation into *SL* is

$$(J \& \sim A) \supset \sim F$$

On the other hand, *this* English sentence, which has the same meaning as the first, is *not* written in canonical logical form.

John and Andrew won’t be friends if John loves Andrew but Andrew doesn’t
love him back.

So we’ll have to say a bit more about how to translate sentences like this into *SL*.

2.4 Translation from English to *SL*

2.4.1 Negation

In English, the word ‘not’ can show up in many places in a sentence. In order for an English sentence to be translated into a wff of *SL* with a ‘ \sim ’, it need not contain the words ‘it is not the case that’. For instance, if we let ‘*H*’ stand in for the English sentence ‘Harry likes chestnuts’, then we may translate the English sentence

Harry doesn’t like chestnuts

as ‘ $\sim H$ ’. The reason is that ‘ $\sim H$ ’ is true if and only if ‘*H*’ is false, and ‘Harry doesn’t like chestnuts’ is true if and only if ‘Harry likes chestnuts’ is false. So our translation has the same meaning as the sentence we wanted to translate. Here’s a more general strategy for translating English sentences into *SL*: re-write the sentences in the *canonical logical form* given by the translation schema from the previous section, and check to see whether the re-written sentence has the same meaning as the sentence that you started out with. If it does, then you may substitute the canonical logical forms for the logical operators of *SL* according to the translation schema of the previous section. If not, then you may not.

For instance, we could re-write ‘Harry doesn’t like chestnuts’ as

It is not the case that Harry likes chestnuts.

Since this contains the canonical logical form ‘it is not the case that’, we may swap this phrase of English out for *SL*’s ‘ \sim ’ to get

\sim Harry likes chestnuts.

We may then use the statement letter ‘*H*’ to represent ‘Harry likes chestnuts’, and we will get the *SL* wff

$\sim H$

A word of warning: just because an English statement contains the word ‘not’, that does not mean that it should be translated into a wff of *SL* with a ‘ \sim ’. In order to see whether it can, we have to see whether re-writing the statement in canonical logical form preserves meaning. For instance, the following sentence contains the word ‘not’:

I hate not getting what I want and I hate getting what I want.

We might attempt to translate this into canonical logical form like so,

It is not the case that I hate getting what I want, and I hate getting what I want.

substitute ‘ \sim ’ for ‘it is not the case that’ and ‘ $\&$ ’ for ‘and’, and get

\sim I hate getting what I want $\&$ I hate getting what I want.

If we then used ‘ H ’ to represent the English ‘I hate getting what I want’, we would get the *SL* wff

$\sim H \& H$

However, *this* wff of *SL* is necessarily false, as the following truth-table shows

H	\sim	H	$\&$	H
T	F	T	F	T
F	T	F	F	F

But the sentence we started with *wasn't* necessarily false. For it is possible that I both hate not getting what I want *and* getting what I want. If this were possible, then I'd hate everything, but surely it's not a logical truth that I don't hate everything. So something went wrong. What went wrong was that ‘I hate not getting what I want’ doesn't have the same meaning as ‘It is not the case that I hate getting what I want’. So we must make sure that translation into canonical logical form preserves meaning in English before we translate that canonical logical form into *SL*.

2.4.2 Conjunction

Many expressions in English have subtle shades of meaning which must be lost when we translate into *SL*. In particular, the following two English expressions will both have the same *SL* translation:

Hannes loves peaches and he loves apples.
Hannes loves peaches but he loves apples.

The second sentence implies some kind of contrast between ‘Hannes loves peaches’ and ‘Hannes loves apples’; whereas the first sentence does not. This subtle difference in

meaning will be lost when we translate into *SL*, since both of these claims are *true* under exactly the same conditions: namely, the condition in which Hannes loves peaches and apples. So, using '*P*' to represent 'Hannes loves peaches' and '*A*' to represent 'Hannes loves apples', they will both be translated into *SL* as '*P* & *A*'.

All of the following expressions of English will also be translated into *SL* with the '&':

$$\left. \begin{array}{l} p \text{ and } q \\ p, \text{ but } q \\ p; \text{ however, } q \\ p, \text{ though } q \\ p \text{ as well as } q \end{array} \right\} \longrightarrow p \& q$$

2.4.3 Disjunction

Both '*p* or *q*' and '*p* unless *q*' are translated into *SL* as '*p* \vee *q*'. If you're unhappy about this translation, think about the following argument: '*p* unless *q*' could be translated as 'If it's not the case that *q*, then *p*', or: ' $\sim q \supset p$ '. And this expression has the very same meaning, in *SL*, as '*p* \vee *q*' (they have the very same truth-table). Thus, '*p* \vee *q*' translates '*p* unless *q*'. If you're *still* unhappy about this translation, think about how you would want to change it (think, that is, about what translation into *SL* you think does a better job than *p* \vee *q*). My guess is that, if you're unhappy with '*p* \vee *q*', then you'll probably be more happy with ' $p \equiv \sim q$ '.

$$\left. \begin{array}{l} p \text{ or } q \\ p \text{ unless } q \end{array} \right\} \longrightarrow p \vee q$$

2.4.4 The Material Conditional and Biconditional

Any of the following English expressions are appropriately translated in SL as ' $p \supset q$ '.

$$\left. \begin{array}{l} \text{If } p, \text{ then } q \\ p \text{ only if } q \\ \text{Only if } q, p \\ q \text{ if } p \\ q, \text{ provided that } p \\ q, \text{ given that } p \\ q \text{ is true whenever } p \text{ is} \\ p \text{ is sufficient for } q \\ q \text{ is necessary for } p \end{array} \right\} \rightarrow p \supset q$$

And any of the following are appropriately translated in SL as ' $p \equiv q$ '.

$$\left. \begin{array}{l} p \text{ if and only if } q \\ p \text{ is necessary and sufficient for } q \\ p \text{ is true when and only when } q \text{ is true} \end{array} \right\} \rightarrow p \equiv q$$

2.5 What a Truth-Table Represents

Let's start with the notion of a *truth-value assignment*.

TRUTH-VALUE ASSIGNMENT. A TRUTH-VALUE ASSIGNMENT is an assignment of truth-value—either true or false—to every statement letter of SL .

Suppose that we don't wish to specify a truth-value assignment completely. That is, we don't wish to specify the truth-values for all of the statement letters of PL . Then, we may choose to just provide a PARTIAL TRUTH-VALUE ASSIGNMENT. A *partial* truth-value assignment merely assigns truth-values to some set of statement letters.

PARTIAL TRUTH-VALUE ASSIGNMENT. A PARTIAL TRUTH-VALUE ASSIGNMENT assigns a truth-value—either true or false—to each statement letter in some set of statement letters.

For instance, a partial truth value assignment, for the set of statement letters $\{A, B, C\}$, is given by saying that A is true, B is false, and C is false.

With these notions under our belt, we can see that the rows of a truth-table are providing us with *every possible* partial truth-value assignment to the statement letters appearing in the wff/argument/set of wffs of *PL* that we're interested in. For instance, if our wff/argument/set of wffs of *PL* contains the statement letters *X*, *Y*, and *Z*, then our truth-table will represent every possible partial truth-value assignment to the set of statement letters $\{X, Y, Z\}$

Every possible partial truth-value assignment to $\{X, Y, Z\}$	<i>X</i>	<i>Y</i>	<i>Z</i>
	<i>T</i>	<i>T</i>	<i>T</i>
	<i>T</i>	<i>T</i>	<i>F</i>
	<i>T</i>	<i>F</i>	<i>T</i>
	<i>T</i>	<i>F</i>	<i>F</i>
	<i>F</i>	<i>T</i>	<i>T</i>
	<i>F</i>	<i>T</i>	<i>F</i>
	<i>F</i>	<i>F</i>	<i>T</i>
	<i>F</i>	<i>F</i>	<i>F</i>

2.6 SL Tautologies, Contradictions, and Contingencies

Recall our definition of *logical tautologies*, *contradictions*, and *CONTINGENCIES*:

Logical Tautology A statement *A* is a LOGICAL TAUTOLOGY iff there is no *possibility* in which *A* is false (*i.e.*, iff *A* is true in every *possibility*).

LOGICAL CONTRADICTION A statement *A* is a LOGICAL CONTRADICTION iff there is no *possibility* in which *A* is true (*i.e.*, iff *A* is false in every *possibility*).

LOGICAL CONTINGENCY A statement *A* is a LOGICAL CONTINGENCY iff there is some *possibility* in which *A* is true and some *possibility* in which *A* is false.

In order to get a rigorous definition of tautologies, contradictions, and contingencies in *SL*, we will simply swap out the notion of a *possibility* with that of a *truth-value assignment*.

SL Tautology A wff of *SL* **A** is an SL TAUTOLOGY iff there is no truth-value assignment on which **A** is false (*i.e.*, iff **A** is true on every truth-value assignment).

SL CONTRADICTION A wff of *SL* **A** is an SL CONTRADICTION iff there is no truth-value assignment on which **A** is true (*i.e.*, iff **A** is false on every truth-value assignment).

SL CONTINGENCY A wff of *SL* **A** is an *SL CONTINGENCY* iff there is some truth-value assignment on which **A** is true and some truth-value assignment on which **A** is false.

For instance, the following truth-table establishes that ' $A \supset (B \supset A)$ ' is an *SL*-tautology:

<i>A</i>	<i>B</i>	<i>A</i>	\supset	$(B \supset A)$		
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>

The following truth-table establishes that ' $\sim (A \vee X) \& A$ ' is an *SL*-contradiction,

<i>A</i>	<i>X</i>	\sim	$(A \vee X)$			$\&$	<i>A</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

And the following truth-table establishes that ' $\sim(L \& M) \equiv (\sim L \& \sim M)$ ' is an *SL*-contingency.

<i>L</i>	<i>M</i>	\sim	$(L \& M)$			\equiv	$(\sim L \& \sim M)$
<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i> <i>T</i> <i>F</i> <i>F</i> <i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i> <i>T</i> <i>F</i> <i>T</i> <i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i> <i>F</i> <i>F</i> <i>F</i> <i>T</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i> <i>F</i> <i>T</i> <i>T</i> <i>F</i>

2.7 SL Equivalence

Recall our definition of logical equivalence:

LOGICAL EQUIVALENCE A pair of statements, *A* and *B*, are **LOGICALLY EQUIVALENT** iff *A* and *B* are true in all the same *possibilities* and false in all the same *possibilities* (i.e., iff it is *impossible* for *A* and *B* to have different truth-values).

Again, in order to get a rigorous definition of equivalence in *SL*, we will simply swap out the notion of a *possibility* with that of a *truth-value assignment*.

SL EQUIVALENCE A pair of wffs of *SL*, **A** and **B**, are *SL-EQUIVALENT* iff **A** and **B** are true in all the same truth-value assignments and false in all the same truth-value assignments (*i.e.*, iff there is no truth-value assignment on which **A** and **B** have different truth-values).

For instance, if we wish to show that ' $A \supset B$ ' and ' $\sim A \vee B$ ' are *SL* equivalent, we may simply show that they have the same truth-values in every row of their joint truth-table, as follows:

A	B	A	\supset	B	\sim	A	\vee	B
T	T	T	T	T	F	T	T	T
T	F	T	F	F	F	T	F	F
F	T	F	T	T	T	F	T	T
F	F	F	T	F	T	F	T	F

2.8 Disjunctive Normal Form and Expressive Completeness

A question to ask yourself: using only the logical operators \sim , $\&$, \vee , \supset , and \equiv , can we write down a wff corresponding to every possible column of truth-values in a truth-table (can we write down a wff which expresses every possible *truth-function*)? That is: is the set of logical operators $\{\sim, \&, \vee, \supset, \equiv\}$ *expressively complete*? Do they allow us to say everything we might wish to say? The answer to this question is 'yes'. In order to prove that this is the answer, we will have to introduce the idea of a wff written in *disjunctive normal form*.

2.8.1 Disjunctive Normal Form

In order to show that we may write down a wff corresponding to any column of truth-values using just the operators in $\{\sim, \&, \vee, \supset, \equiv\}$, we will show something stronger: that we may write down a wff corresponding to any column of truth-values using just the operators in $\{\sim, \&, \vee\}$. That is: we will show that $\{\sim, \&, \vee\}$ is an expressively complete set of logical operators.

To see how we may do this, consider the following truth-function:

A	B	
T	T	F
T	F	T
F	T	F
F	F	T

This truth-function gives us true iff either A is true and B is false or if A and B are both false. So if we wish to write down a wff of SL that expresses this truth-function, then we must write down a wff of SL that is only true in the second and fourth rows of this truth table. Consider first the wff ' $A \& \sim B$ '. This wff is true only in the second row of the truth table.

A	B	$A \& \sim B$
T	T	F
T	F	T
F	T	F
F	F	F

Next consider the wff ' $\sim A \& \sim B$ '. This wff is true only in the fourth row of the truth-table.

A	B	$\sim A \& \sim B$
T	T	F
T	F	F
F	T	F
F	F	T

Now, if we disjoin these two wffs, we will get a wff of SL that is true in only the second and the fourth rows of the truth-table.

A	B	$(A \& \sim B) \vee (\sim A \& \sim B)$
T	T	F
T	F	T
F	T	F
F	F	T

And this is precisely the truth-function we wished to get.

The wff ' $(A \& \sim B) \vee (\sim A \& \sim B)$ ' is written in *disjunctive normal form*. Here is a precise definition of what it is for a wff to be in *disjunctive normal form*:

A wff of SL is in DISJUNCTIVE NORMAL FORM iff it is a disjunction* of conjunctions*, the conjuncts* of which are either atomic formulae or negated atomic formulae.

In this definition, we are using the terms ‘disjunction’ and ‘conjunction’ in a slightly more liberal sense than the way we have been using them up to this point. To flag this, I have added asterisks to those terms to distinguish them from the stricter definitions provided earlier. The differences are these: A disjunction* may have only one disjunct*—so that, e.g., ‘ $(F \& G)$ ’ counts as a disjunction with just one disjunct*. Similarly, a conjunction* may have only one conjunct*—so that, e.g., ‘ $A \vee \sim A$ ’ counts as a disjunction of conjunction*, since ‘ A ’ and ‘ $\sim A$ ’ both count as conjunctions* with only one conjunct. Thus, ‘ A ’ is also in disjunctive normal form and ‘ $\sim A$ ’ is also in disjunctive normal form.

Now we may show that, for any column of truth-values in any truth-table, we may construct a corresponding wff of SL which has just that column of truth-values. Here’s a simple recipe for doing so:

1. If the column of truth-values has no ‘ T ’s (if it is an SL contradiction), then write ‘ $A \& \sim A$ ’.
2. If the column of truth-values has no ‘ F ’s (if it is an SL tautology), then write ‘ $A \vee \sim A$ ’.
3. If the column of truth-values has some ‘ T ’s and some ‘ F ’s, then,
 - (a) for every row with a ‘ T ’, write down a conjunction of (negations of) the atomic formulae in that row.
 - i. If an atomic formula has the truth-value ‘ T ’ in that row, then do not negate it in the conjunction.
 - ii. If an atomic formulae has the truth-value ‘ F ’ in that row, then negate it in the conjunction.
 - (b) Disjoin all the conjunctions that you wrote down in step 3(a).

For instance, consider the following truth-function:

<i>B</i>	<i>C</i>	<i>D</i>	
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>F</i>	T
<i>F</i>	<i>T</i>	<i>T</i>	T
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	T
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

This truth-function is true in rows 4, 5, and 7. So we will require 3 disjuncts. In row 4, the atomic formula *B* is true, the atomic formula *C* is false, and the atomic formulae *D* is false. So we should write down the conjunction

$$B \& (\sim C \& \sim D)$$

(Of course, the conjunction ' $(B \& \sim C) \& \sim D$ ' would have worked just as well, as would ' $\sim C \& (\sim D \& B)$ '.) In row 5, the atomic formula *B* is false, the atomic formula *C* is true, and the atomic formula *D* is true. So we should write down the conjunction:

$$\sim B \& (C \& D)$$

(Again, we could change the order of the conjuncts or the order of the parentheses, and this wouldn't matter.) Finally, in row 7, *B* is false, *C* is false, and *D* is true. So we should write down the conjunction

$$\sim B \& (\sim C \& D)$$

(Again, the order of the conjuncts doesn't matter.) Finally, we disjoin these three conjunctions to get:

$$[B \& (\sim C \& \sim D)] \vee \{[\sim B \& (C \& D)] \vee [\sim B \& (\sim C \& D)]\}$$

Now we have a wff of *SL* expressing the same truth-function we started with:

<i>B</i>	<i>C</i>	<i>D</i>	$[B \ \& \ (\sim C \ \& \ \sim D)]$	\vee	$\{[\sim B \ \& \ (C \ \& \ D)]$	\vee	$[\sim B \ \& \ (\sim C \ \& \ D)]\}$
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	F	<i>F</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	F	<i>F</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	F	<i>F</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	T	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	T	<i>T</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	F	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	T	<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	F	<i>F</i>	<i>F</i>	<i>F</i>

Note that, for any series of conjunctions, the order of the conjuncts doesn't matter, nor does it matter where we put the parentheses. Similarly, for any series of disjunctions, the order of the disjuncts doesn't matter, nor does it matter where we put the parentheses. So, we may afford ourselves the additional notational convention of omitting these parentheses, writing the wff above as simply:

$$(B \ \& \ \sim C \ \& \ \sim D) \vee (\sim B \ \& \ C \ \& \ D) \vee (\sim B \ \& \ \sim C \ \& \ D)$$

There is some ambiguity in the wff above, but it is not ambiguity that affects the meaning of the wff. Every way of disambiguating yields the same truth-function. So this is a harmless ambiguity.

2.8.2 Expressive Completeness

We've shown that we can write down any truth-function using just the logical operators \sim , \vee , and $\&$. That is: we've shown that the set of logical operators $\{\sim, \vee, \&\}$ is *expressively complete*. But we can show more than this. We can show that the set $\{\sim, \vee\}$ is expressively complete.

Here's how: suppose that we have a wff of the form $\lceil \mathbf{P} \ \& \ \mathbf{Q} \rceil$. Then, replace this wff with one of the form $\lceil \sim (\sim \mathbf{P} \ \vee \ \sim \mathbf{Q}) \rceil$. The truth-table schema below shows that these two wffs are *SL*-equivalent:

P	Q	P & Q	\sim	$(\sim\mathbf{P} \vee \sim\mathbf{Q})$
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>

Now, we can go through any wff in disjunctive normal form and replace any subformula of the form ' $\mathbf{P} \& \mathbf{Q}$ ' with an *SL*-equivalent subformula which utilizes only ' \mathbf{P} ', ' \mathbf{Q} ', ' \sim ', and ' \vee '. If we do this for every subformula in the wff, then we will have removed all of the '&'s. And if we can do this for every wff in disjunctive normal form, then we can express every truth-function using just \sim and \vee . So, if we can do this, then $\{\sim, \vee\}$ must be expressively complete.

For instance, consider the following wff, which is in disjunctive normal form:

$$(R \& S) \vee (\sim R \& \sim S)$$

In this, there are two subformulae whose main operators are '&': ' $R \& S$ ' and ' $\sim R \& \sim S$ '. The above schema tells us that we may replace ' $R \& S$ ' with ' $\sim (\sim R \vee \sim S)$ ', and that we may replace ' $\sim R \& \sim S$ ' with ' $\sim (\sim \sim R \vee \sim \sim S)$ ', and we will get something which is *SL*-equivalent to the thing that we started out with. So

$$\sim(\sim R \& \sim S) \vee \sim(\sim \sim R \vee \sim \sim S)$$

is *SL*-equivalent to $(R \& S) \vee (\sim R \& \sim S)$. And this wff utilizes only the logical operators \sim and \vee .

The schema above assures us that we may do this for *any* wff in disjunctive normal form. And we know that, for any truth-function, there is some wff in disjunctive normal form which expresses that truth-function. So we know that we may express any truth-function using just the operators \sim and \vee . So $\{\sim, \vee\}$ is an expressively complete set of operators.

We may also show that $\{\sim, \&\}$ is an expressively complete set of operators (you'll be asked to show this in the in class exercise for today).

The Scheffer Stroke

Is there a single logical operator that is expressively complete, all by itself? The answer is 'yes'. Actually, there's more than one. The logical operator ' \downarrow ', known as the 'Scheffer stroke', and the logical operator ' \uparrow ', known as 'Pierce's arrow', are both expressively

complete. These logical operators are defined by the following truth-tables:

P	Q	P Q
<i>T</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>T</i>

P	Q	P ↓ Q
<i>T</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>

We already know that $\{\sim, \vee\}$ is expressively complete (from the in class exercise). From this, we can show that $\{| \}$ is expressively complete by showing how to replace every wff of the form $\lceil \sim \mathbf{P} \rceil$ with an *SL*-equivalent wff which utilizes only $\lceil \mathbf{P} \rceil$ and $\lceil | \rceil$, and showing how to replace every wff of the form $\lceil \mathbf{P} \& \mathbf{Q} \rceil$ with an *SL*-equivalent wff which utilizes only $\lceil \mathbf{P} \rceil$, $\lceil \mathbf{Q} \rceil$, and $\lceil | \rceil$. This is accomplished by the following truth-table schemas.

P	~P	P P
<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>

P	Q	P & Q	(P Q)	 	(P Q)
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>

This shows that $\{| \}$ is expressively complete. On your problem set, you'll be asked to show that $\{\downarrow\}$ is expressively complete as well.

2.9 SL-Validity and SL-Invalidity

To begin with, a bit of new notation: If we have an argument from the premises **P**, **Q**, **R** to the conclusion **S**, then, rather than writing this as we have been, like so

$$\frac{\begin{array}{c} \mathbf{P} \\ \mathbf{Q} \\ \mathbf{R} \end{array}}{\mathbf{S}}$$

we will denote the argument by putting single forward slashes between premises, and putting a double forward slash between the premises and the conclusion, like so:

$$\mathbf{P / Q / R // S}$$

If we have a collection of wffs of *SL*, one of which is designated the conclusion, the others of which are designated premises, then we have what we will call a '*SL*-argument'.

A *SL*-ARGUMENT is a collection of wffs of *SL*, one of which is designated the conclusion, and the others of which are designated the premises.

Recall the definition of DEDUCTIVE VALIDITY. An argument is deductively valid if and only if there is no *possibility* in which all of the premises are true but the conclusion is false. Equivalently: an argument is deductively valid if and only if every possibility in which the premises of the argument are all true is a possibility in which the conclusion is true also. To model deductive validity in the language *SL*, we will give a definition of validity within the language *SL*—what we will call '*SL*-VALIDITY'—which is just the same as the definition of deductive validity, except with a formal substitution for the notion of a *possibility*. The substitution we will make is this: for '*possibility*', we will substitute 'truth-value assignment'.

A *SL*-argument is *SL*-VALID if and only if there is no truth-value assignment in which all of the premises are true and the conclusion is false.

Because the truth-values of the wffs of *SL* appearing in a *SL*-argument are determined entirely by the statement letters appearing in those wffs, we need not consider *every* truth-value assignment. Rather, it will be enough to look at all the *partial* truth-value assignments to those statement letters appearing in the *SL*-argument. Each such partial truth-value assignment corresponds to a row of the truth-table for the *SL*-argument. So, another, equivalent, definition of *SL*-validity is this:

A *SL*-argument is *SL*-VALID if and only if, in the argument's truth-table, there is no row in which the premises of the argument are all true and the conclusion is false.

So, for instance, suppose we wish to determine whether the following *SL*-argument is *SL*-valid:

$$\sim (P \& Q) / P // \sim Q$$

To check, we first construct the truth-table for the argument.

<i>P</i>	<i>Q</i>	
<i>T</i>	<i>T</i>	
<i>T</i>	<i>F</i>	
<i>F</i>	<i>T</i>	
<i>F</i>	<i>F</i>	

Next, we place the argument's premise in one column, and the argument's conclusion in another, like so, and, underneath each of the statement letters, we copy over the truth-values from the first two columns:

<i>P</i>	<i>Q</i>	$\sim (P \ \& \ Q)$	<i>P</i>	$\sim Q$
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

(Here, I've placed two vertical lines between the premise and the conclusion just to indicate that the conclusion is the thing to the right of those double vertical lines.)

We then finish up by, in the case of the argument's premise ' $\sim (P \ \& \ Q)$ ', determining the appropriate column of truth values beneath $(P \ \& \ Q)$, and then determining the appropriate column of truth values beneath $\sim (P \ \& \ Q)$. For the argument's conclusion, $\sim Q$, we determine the appropriate column of truth-values beneath $\sim Q$. When we're done, we place a box around the columns beneath the premises' and the conclusion's main operators.

<i>P</i>	<i>Q</i>	\sim (<i>P</i> & <i>Q</i>)	<i>P</i>	\sim <i>Q</i>
<i>T</i>	<i>T</i>	<i>F</i> <i>T</i> <i>T</i> <i>T</i>	<i>T</i>	<i>F</i> <i>T</i>
<i>T</i>	<i>F</i>	<i>T</i> <i>T</i> <i>F</i> <i>F</i>	<i>T</i>	<i>T</i> <i>F</i>
<i>F</i>	<i>T</i>	<i>T</i> <i>F</i> <i>F</i> <i>T</i>	<i>F</i>	<i>F</i> <i>T</i>
<i>F</i>	<i>F</i>	<i>T</i> <i>F</i> <i>F</i> <i>F</i>	<i>F</i>	<i>T</i> <i>F</i>

Now, in order to decide whether the argument is *SL*-valid or *SL*-invalid, we need to determine whether every row in which the premise is true is a row in which the conclusion is true also. The first premise is true in rows 2–4, and the second premise is true in rows 1 and 2. Therefore, both premises are only true in row 2:

P	Q	\sim	$(P \ \& \ Q)$			P	\sim	Q
T	T	F	T	T	T	T	F	T
T	F	T	T	F	F	T	T	F
F	T	T	F	F	T	F	F	T
F	F	T	F	F	F	F	T	F

And, in row 2, the conclusion is true also. So, there is no row of the truth-table in which the premises are all true yet the conclusion is false. So, the *SL*-argument

$$\sim (P \ \& \ Q) / P \ // \ \sim Q$$

is *SL*-valid.

A *SL*-argument is *SL*-invalid if and only if it is not *SL*-valid. Thus:

A *SL*-argument is *SL*-INVALID if and only if there is some truth-value assignment on which all of the argument's premises true yet its conclusion is false.

Or, equivalently:

A *SL*-argument is *SL*-INVALID if and only if there is some row of its truth-table in which all of the premises are true and in which the conclusion is false.

Suppose that we want to show that the following *SL*-argument is *SL*-invalid:

$$A \supset C / \sim A \ // \ \sim C$$

Then, we may construct the truth-table for this *SL*-argument. We will arrive at the following:

A	C	A	\supset	C	\sim	A	\sim	C
T	T	T	T	T	F	T	F	T
T	F	T	F	F	F	T	T	F
F	T	F	T	T	T	F	F	T
F	F	F	T	F	T	F	T	F

Both of the premises are true in rows 3 and 4 of the truth-table. So we restrict our attention to those rows. If the conclusion is also true in both of those rows of the truth table, then the argument is *SL*-valid. If, however, the conclusion is false in one of those rows of the truth-table, then the argument is *SL*-invalid.

<i>A</i>	<i>C</i>	<i>A</i>	\supset	<i>C</i>	\sim	<i>A</i>	\sim	<i>C</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>	T	<i>T</i>	T	<i>F</i>	F	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>

Look, however, at the third row of the truth-table. On this row of the truth-table, the premises of the argument are true, yet its conclusion is false. So the argument is *SL*-invalid.

Just as we saw with deductive validity and deductive invalidity, we can give an equivalent definition of *SL*-validity and *SL*-invalidity by introducing the notion of a *SL*-counterexample.

A *SL*-COUNTEREXAMPLE to the *SL*-validity of a *SL*-argument is a truth-value assignment on which the premises of the argument are all true, yet the conclusion is false.

Alternatively:

A *SL*-COUNTEREXAMPLE to the *SL*-validity of a *SL*-argument is a row of the argument's truth table in which all of the premises are true and the conclusion is false.

Now, a *SL*-argument is *SL*-valid if and only if it has no *SL*-counterexample.

A *SL*-argument is *SL*-VALID if and only if it has no *SL*-counterexample.

And, thus, a *SL*-argument is *SL*-invalid if and only if it has a *SL*-counterexample.

A *SL*-argument is *SL*-INVALID if and only if it has a *SL*-counterexample.

For instance, the *SL*-argument

$$A \supset C / \sim A // \sim C$$

considered above has the following *SL*-counterexample:

A is false and *C* is true

(This is the assignment of truth-values to A and C which corresponds to the third row of the truth-table above.)

2.10 SL-Validity and SL-Tautologies

The *SL*-argument

$$\sim (P \& Q) / P // \sim Q$$

is *SL*-valid, as we saw above. Note also that the wff

$$(\sim(P \& Q) \& P) \supset \sim Q$$

is an *SL*-tautology:

P	Q	$(\sim$	$(P$	$\&$	$Q)$	$\&$	$P)$	\supset	\sim	Q
T	T	F	T	T	T	F	T	T	F	T
T	T	T	T	F	F	T	T	T	T	F
T	T	T	F	F	T	F	F	T	F	T
T	T	T	F	F	F	F	F	T	T	F

This isn't an accident. In general, an *SL*-argument $\lceil \mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C} \rceil$ is *SL*-valid if and only if the material conditional $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ is an *SL*-tautology. (Here, I've omitted the parentheses around the conjuncts of the antecedent since it doesn't matter how we group together the conjuncts; it won't change the truth-function they determine.)

The *SL*-argument

$$\mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C}$$

is *SL*-valid if and only if

$$(\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C}$$

is an *SL*-tautology.

To see why, think about the circumstances in which an argument is *SL*-valid. It is *SL*-valid iff, in every row of the argument's truth table in which the premises are *all* true, the conclusion is true also. It doesn't matter whether the conclusion is true when *some*, but not *all* of the premises are true. The only rows of the truth table we have to consider are those in which *all* of the premises are true. But the rows of the truth-table

in which *all* of the premises are true are just those rows of the truth table in which the *conjunction* of all of the premises, $\lceil \mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N \rceil$ is true. In any row of the truth table in which not all of the premises are true, this conjunction is false, and therefore, the material conditional $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ is automatically true (since a material conditional is true whenever its antecedent is false). In those rows of the truth-table in which *all* of the premises are true, the conjunction of all of the premises $\lceil \mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N \rceil$ will be true also. If the conclusion $\lceil \mathbf{C} \rceil$ is true in all of these rows, then the material conditional $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ will be true in those rows as well. So, if $\lceil \mathbf{C} \rceil$ is true in every row that the premises are all true, then $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ will be true in every row or its truth-table. So $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ will be a tautology.

If, on the other hand, the *SL* argument $\lceil \mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C} \rceil$ is invalid, then there will be some row of its truth table in which all of the premises are true, yet the conclusion is false. But then there will be some row of the truth-table for $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ in which the conjunction $\lceil \mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N \rceil$ is true yet $\lceil \mathbf{C} \rceil$ is false. So there will be some row of the truth table in which $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ is false. So $\lceil (\mathbf{P}_1 \& \mathbf{P}_2 \& \dots \& \mathbf{P}_N) \supset \mathbf{C} \rceil$ will not be an *SL*-tautology.

2.11 *SL*-Consistency & *SL*-Inconsistency

Suppose that we've got an arbitrary set of wffs of *SL*. For instance, suppose that we've got the following set:

$$\left\{ \begin{array}{l} A \supset \sim B \\ B \supset A \\ B \end{array} \right\}$$

This set contains three wffs of *SL*: first, ' $A \supset \sim B$ ', second, ' $B \supset A$ ', and third, ' B '.

For an arbitrary set of wffs of *SL* like this, it could either be the case that:

1. There is some truth-value assignment on which every wff in the set is true; or
2. There is no truth-value assignment on which every wff in the set is true.

In case (1), there's some way of assigning truth-values to the statement letters of *SL* such that you can make every wff in the set true at once. In that case, we say that the set of wffs of *SL* is *SL-consistent*.

A set of wffs of *SL* is *SL-CONSISTENT* if and only if there is some truth-value assignment on which all of the wffs are true.

Or, equivalently,

A set of wffs of *SL* is *SL*-CONSISTENT if and only if there is some row of their joint truth table in which all of the wffs are true.

For instance, consider the following set of wffs of *SL*:

$$\left\{ \begin{array}{l} A \vee B \\ \sim B \& A \\ A \supset \sim B \end{array} \right\}$$

When we construct their joint truth-table, we find:

<i>A</i>	<i>B</i>	<i>A</i>	\vee	<i>B</i>	\sim	<i>B</i>	$\&$	<i>A</i>	<i>A</i>	\supset	\sim	<i>B</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	T	<i>F</i>	<i>T</i>	<i>F</i>	T	<i>T</i>	<i>T</i>	T	<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>

Each of these three wffs of *SL* are true on line 2 of the truth-table. So, this set of wffs is *SL*-consistent. They are capable of all three being true on the very same (partial) truth-value assignment—namely, the partial truth-value assignment ‘*A* is true and *B* is false’.

A set of wffs of *SL* is *SL*-inconsistent if and only if there is *no* truth-value assignment which makes all of them true at once.

A set of wffs of *SL* is *SL*-INCONSISTENT if and only if there is no truth-value assignment on which all of the wffs are true (i.e., if and only if, on every truth-value assignment, at least one of the wffs in the set is false).

Or, equivalently,

A set of wffs of *SL* is *SL*-INCONSISTENT if and only if there is no row of their joint truth table in which all of the wffs are true (i.e., if and only if, in every row of their truth table, at least one of the wffs in the set is false).

Logical Property	Applies Only To
<i>SL</i> -Validity	<i>SL</i> -Arguments
<i>SL</i> -Invalidity	<i>SL</i> -Arguments
<i>SL</i> -Tautology	individual wffs of <i>SL</i>
<i>SL</i> -Self-Contradiction	individual wffs of <i>SL</i>
<i>SL</i> -Contingency	individual wffs of <i>SL</i>
<i>SL</i> -Equivalent	pairs of wffs of <i>SL</i>
<i>SL</i> -Consistent	sets of wffs of <i>SL</i>
<i>SL</i> -Inconsistent	sets of wffs of <i>SL</i>

Figure 2.1: : The logical properties of *SL* and the kinds of entities to which they apply.

For instance, consider the set of wffs with which we began this section:

$$\left\{ \begin{array}{l} A \supset \sim B \\ B \supset A \\ B \end{array} \right\}$$

The joint truth-table for these three wffs of *SL* is shown below:

<i>A</i>	<i>B</i>	<i>A</i>	\supset	\sim	<i>B</i>	<i>B</i>	\supset	<i>A</i>	<i>B</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>

In the first row of the truth-table, $A \supset \sim B$ is false; in the second row of the truth-table, *B* is false; in the third row of the truth-table, $B \supset A$ is false; and in the fourth row of the truth-table, *B* is false. So one of the three wffs is false in every row of their truth-table. So, there's no row of their truth-table on which they are *all* true. So this set of wffs is *SL-inconsistent*.

2.12 *SL*-Consistency and the Other Logical Properties of *SL*

The first three logical properties we encountered—*SL*-tautology, *SL*-self-contradiction, and *SL*-contingency—applied to individual wffs of *SL*. The next logical property—*SL*-equivalence—applied to *pairs* of wffs of *SL*. The next two logical properties—*SL*-validity

and *SL*-invalidity— applied to *arguments* of *SL*. The final two logical properties—*SL*-consistency and *SL*-inconsistency—apply to arbitrary *sets* of wffs of *SL*.

It's important to keep in mind the kinds of entities to which these logical properties apply. I have summarized this information for you in figure 6.4. You should resist any urge to say, for instance, that an individual wff of *SL* is '*SL*-valid', or that an argument of *SL* is a *SL*-tautology. These claims would be false; for, given the definitions provided here, only *SL*-arguments can have the property being *SL*-valid, and only individual wffs of *SL* can be *SL*-tautologies.

Nevertheless, we *can* say some interesting things about how the 4 families of logical properties displayed in figure 6.4 are related to one another. Thinking through some of these relationships can bring us to a deeper understanding of what these properties amount to. In fact, there is a rather deep relationship between these four families of properties—given any *one* of these families of properties, we can *define* all the rest of the properties in terms of them.

I won't show this for all of the families of properties (though you should think through, for yourself, how to show it); but it will be instructive to walk through how to reduce all of the other logical properties we've learned about to the notions of *SL*-consistency and *SL*-inconsistency.

To being with, we can define the notion of an *SL* tautology in terms of *SL*-consistency, as follows:

A wff of *SL* $\lceil \mathbf{P} \rceil$ is an *SL*-tautology if and only if the set $\{\sim \mathbf{P}\}$ is *SL*-inconsistent.

To see why this is so, think about the circumstances in which $\lceil \mathbf{P} \rceil$ is an *SL*-tautology. It is an *SL*-tautology iff it is true in every row of its truth-table. But, if $\lceil \mathbf{P} \rceil$ is true in every row of its truth-table, then $\lceil \sim \mathbf{P} \rceil$ will be false in every row of its truth-table. But if $\lceil \sim \mathbf{P} \rceil$ is false in every row of its truth-table, then there will be no row on which all of the members of the set $\{\sim \mathbf{P}\}$ are true—since $\{\sim \mathbf{P}\}$ has only one member, that means that there is no row of the truth table on which that one member is true. So, if $\lceil \mathbf{P} \rceil$ is an *SL*-tautology, then $\{\sim \mathbf{P}\}$ will be *SL*-inconsistent. And, if $\{\sim \mathbf{P}\}$ is *SL*-inconsistent, then there's no row of the truth-table on which $\lceil \sim \mathbf{P} \rceil$ is true. But then, by the definition of ' \sim ', then will be no row of the truth-table on which $\lceil \mathbf{P} \rceil$ is false. So $\lceil \mathbf{P} \rceil$ will be true in every row of its truth-table. So $\lceil \mathbf{P} \rceil$ will be an *SL*-tautology.

We can similarly define the notion of an *SL*-contradiction in terms of inconsistency by saying that $\lceil \mathbf{P} \rceil$ is an *SL*-contradiction iff the set $\{\mathbf{P}\}$ is *SL*-inconsistent.

A wff of SL $\lceil \mathbf{P} \rceil$ is an SL-contradiction if and only if the set $\{\mathbf{P}\}$ is SL-inconsistent.

For, $\lceil \mathbf{P} \rceil$ is an SL-contradiction iff it is false in every row of its truth table, which is so iff there's no row of the truth-table on which every member of $\{\mathbf{P}\}$ is true.

Finally, we can define the notion of an SL-contingency in terms of SL-consistency, as follows:

A wff of SL $\lceil \mathbf{P} \rceil$ is an SL-contingency if and only if both $\{\mathbf{P}\}$ and $\{\sim\mathbf{P}\}$ are SL-consistent.

For, if and only if $\{\mathbf{P}\}$ is SL-consistent, there's some row of the truth-table on which \mathbf{P} is true, and if and only if $\{\sim\mathbf{P}\}$ is SL-consistent, there's some row of the truth-table on which \mathbf{P} is false. So there's some row of the truth-table on which $\lceil \mathbf{P} \rceil$ is true and some row on which it is false. So it's an SL-contingency.

We can go on to talk about the notion of SL-equivalence by noting that, if $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are SL-equivalent, then there's no row of the truth-table on which $\lceil \mathbf{P} \rceil$ is true and $\lceil \mathbf{Q} \rceil$ is false, and there's no row of the truth-table on which $\lceil \mathbf{P} \rceil$ is false and $\lceil \mathbf{Q} \rceil$ is true. So every row of the truth-table is one on which either both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are true or both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are false. But those are exactly the circumstances in which $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is true. So $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are SL-equivalent iff $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is true in every row of its truth-table—that is to say, iff $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is an SL-tautology. But we already know how to make sense of SL-tautologies in terms of inconsistency. So we can say:

$\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are SL-equivalent iff $\{\sim(\mathbf{P} \equiv \mathbf{Q})\}$ is SL-inconsistent.

Finally, we can make sense of the notion of SL-validity in terms of SL-consistency by noting that an argument of SL, $\lceil \mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C} \rceil$ is SL-valid if and only if there's no row of the truth-table on which $\lceil \mathbf{P}_1 \rceil$, $\lceil \mathbf{P}_2 \rceil$, ..., and $\lceil \mathbf{P}_N \rceil$ are all true yet $\lceil \mathbf{C} \rceil$ is false. Since $\lceil \mathbf{C} \rceil$ is false iff $\lceil \sim\mathbf{C} \rceil$ is true, we can re-write this as follows: there is no row of the truth table on which $\lceil \mathbf{P}_1 \rceil$, $\lceil \mathbf{P}_2 \rceil$, ..., $\lceil \mathbf{P}_N \rceil$, and $\lceil \sim\mathbf{C} \rceil$ are all true. But *that's* so iff the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim\mathbf{C}\}$ is SL-inconsistent. Thus:

$\lceil \mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C} \rceil$ is SL-valid iff $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim\mathbf{C}\}$ is SL-inconsistent.

And since an argument is SL-invalid iff there's *some* row of the truth-table on which all

of the premises are true yet the conclusion is false, an argument is *SL*-invalid iff there's some row of the truth-table on which $\lceil \mathbf{P}_1 \rceil$, $\lceil \mathbf{P}_2 \rceil$, ..., $\lceil \mathbf{P}_N \rceil$, and $\lceil \sim \mathbf{C} \rceil$ are all true. Thus:

$\lceil \mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C} \rceil$ is *SL*-invalid iff $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim \mathbf{C}\}$ is *SL*-consistent.

Chapter 3

Sentence Logic Trees

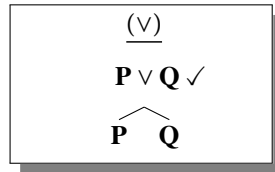
3.1 Truth Trees: The Basics

Today, we're going to learn about a procedure for testing the consistency of sets of wffs of *SL*. Because we may define all of the other logical notions of *SL* in terms of consistency (as we saw last time), this will allow us to test whether wffs of *SL* are *SL*-tautologies, *SL*-contingencies, or *SL*-contradictions; it will allow us to test whether pairs of wffs of *SL* are *SL*-equivalent; and it will allow us to test whether an *SL*-argument is *SL*-valid or *SL*-invalid.

In broad outline, here's how the procedure is going to work: we're going to start by placing all of the wffs we're interested in in a vertical stack. If, for instance, we're interested in the set of wffs $\{A \vee \sim B, \sim A, B\}$, then we will begin with the following stack of wffs:

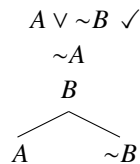
$$\begin{array}{c} A \vee \sim B \\ \sim A \\ B \end{array}$$

These wffs form the trunk of a tree. Once we've written down the wffs on the trunk, we will begin by applying the relevant tree rules to those wffs on the tree which are not atomic or negations of atomic formulae. For instance, in the set of wffs above, $\sim A$ is the negation of an atomic sentence, so we need not apply any rule to it; B is an atomic sentence, so we need not apply any rule to it. ' $A \vee \sim B$ ' *isn't* an atomic sentence, so we *will* need to apply a rule to it. For each wff which is neither atomic nor a negation of an atomic wff, there is one unique rule for that wff. This rule is determined by the syntactic structure of the wff we're considering. For instance, with ' $A \vee \sim B$ ', the main operator is the wedge, \vee , so we will use the rule associated with the wedge, which is:



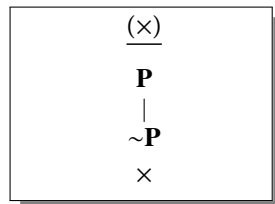
Here's how to read this rule: it says that, if you have a wff of the form $\mathbf{P} \vee \mathbf{Q}$ appearing at some point in the tree, then you may place a check next to that wff; when you do so, you must, on every open branch of the tree below that wff, create two new branches. On one of those branches, you must write \mathbf{P} ; on the other branch, you must write \mathbf{Q} . Once you have done this for every open branch beneath the wff $\mathbf{P} \vee \mathbf{Q}$, you are finished with that wff. The checkmark reminds you that you have already applied the relevant rule to that wff; and, once a rule has been applied to a wff once, it cannot be applied to that wff again.

So, for instance, in the tree we started above, when we apply the rule (\vee) to $A \vee \sim B$, we get the following tree:



Before, when I was explaining the rule (\vee) , I used the notion of an *open branch*. The tree above now has two branches: the one going off to the left and ending in A , and the one going off to the right and ending in $\sim B$. On the first branch, the wffs $A \vee \sim B$, $\sim A$, B , and A appear. On the second branch, the wffs $A \vee \sim B$, $\sim A$, B , and $\sim B$ appear. Note that the wffs on the trunk of the tree appear on both branches.

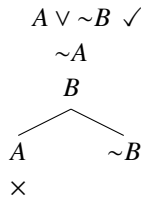
In order to explain what I mean by an *open branch* of the tree, let me introduce one more rule:



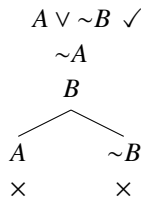
This rule says that, anytime you have a wff of the form \mathbf{P} and a wff of the form $\sim \mathbf{P}$ appearing on the same branch of the tree—that is, any time you have a formula and its negation appearing on the same branch of the tree—you should mark an \times at the bottom

of that branch. When you do so, that branch of the tree has *closed*. Once a branch has closed, it may no longer be extended. The rules only allow us to extend *open* branches. A branch is *open* iff it is not closed.

On the tree above, the first branch has both the wff ' $\sim A$ ' and the wff ' A '. So, the rule (\times) tells us that we must close this branch by writing ' \times ' at the bottom of the branch, as follows:



Similarly, the second branch has both the wff ' B ' and the wff ' $\sim B$ '. So, the rule (\times) tells us that we must close this branch, too.



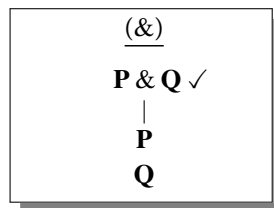
Now, all branches have closed. If all the branches of a tree have closed, then you are done with that tree, and we say that *the tree closes*. There is one other way to finish a tree. If you have checked off and applied all of the relevant rules to the wffs appearing on the open branches of that tree which are not atomic wffs or negations of atomic wffs, then you are done with that tree. If, after having applied all of those rules, at least one branch of the tree remains open, then we say that *the tree does not close*, or that *the tree remains open*.

To complete a tree:

1. Apply the relevant rules to all wffs appearing on open branches, in any order you like.
2. If a wff $\lceil P \rceil$ and its negation $\lceil \sim P \rceil$ appear on the same branch, then close that branch by writing '×' at the bottom of the branch.
3. If every branch closes, then you are done; in this case, we say that *the tree closes*.
4. If you have applied every relevant rule to every wff on every open branch which is not atomic or the negation of an atomic wff, then you are done; if, after doing this, there remains an open branch, then we say that *the tree remains open*.

3.2 Rules for Truth Trees

There are ten rules for truth trees. We have already learned about (\vee) , for wffs of the form $\lceil P \vee Q \rceil$. Next is the rule for wffs of the form $\lceil P \& Q \rceil$.

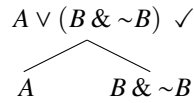


This rule says that, if you have a wff of the form $\lceil P \& Q \rceil$ appearing on an open branch, then you may place a check next to that wff; when you do so, you must, on every open branch of the tree on which the wff $\lceil P \& Q \rceil$ appears, write $\lceil P \rceil$ and, immediately beneath it, $\lceil Q \rceil$.

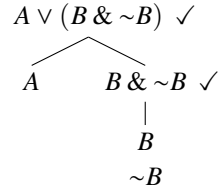
For instance, suppose that we begin with the following wff at the trunk of the tree:

$$A \vee (B \& \sim B)$$

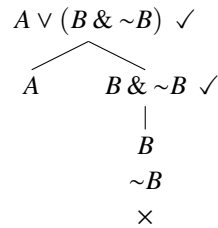
There is one wff in this tree, so we must apply the rule (\vee) to it—since it is a disjunction. When we do so, we get:



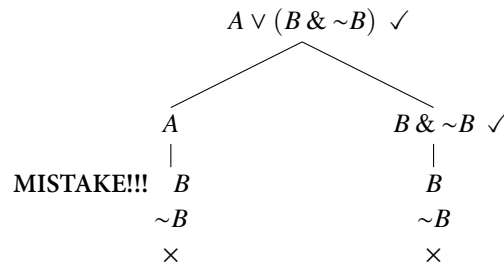
Now we must apply the rule ($\&$) to the wff ' $B \& \sim B$ '—since this is a wff of the form ' $\mathbf{P} \& \mathbf{Q}$ '. When we do so, we get:



Since both ' B ' and ' $\sim B$ ' appear on the second branch of the tree, we must, by rule (\times), close this branch.

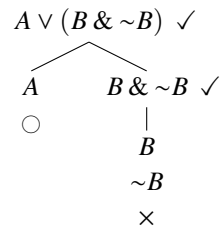


Note that, when I applied the rule ($\&$), I only wrote B and $\sim B$ to the branch on the left. I didn't, for instance, do this:



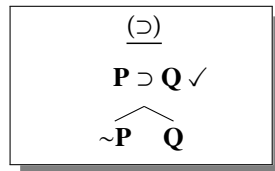
Why not? Because the rule ($\&$) says that you should only write ' \mathbf{P} ' and ' \mathbf{Q} ' to the end of those branches *on which the wff ' $\mathbf{P} \& \mathbf{Q}$ ' appears*. But ' $B \& \sim B$ ' does not appear on the first branch. So you should not write ' B ' and ' $\sim B$ ' on that branch.

Once we have applied the rule (\vee) to the wff ' $A \vee (B \& \sim B)$ ' and the rule ($\&$) to the wff ' $B \& \sim B$ ', there is only the atomic formula ' A ' left on the first branch of the tree. So there are no more rules to be applied. So we are done with the tree, and the first branch has remained open. To indicate this, we may write a circle, ' \circ ' at the end of that branch.



This tells us that the tree has remained open.

The next rule is for wffs of the form $\lceil P \supset Q \rceil$,



This rule tells us that, if we have a wff of the form $\lceil P \supset Q \rceil$ on an open branch of the tree, then, we may place a checkmark next to this wff and split every open branch on which that wff lies in two. On the left-hand-side of this split, we should write $\lceil \sim P \rceil$, and on the right-hand-side, we should write $\lceil Q \rceil$.

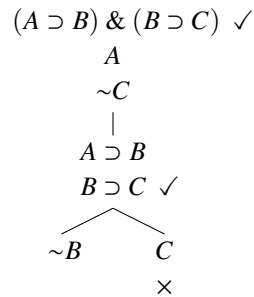
For instance, suppose that we have a tree with the following trunk:

$$\begin{array}{c}
 (A \supset B) \& (B \supset C) \\
 A \\
 \sim C
 \end{array}$$

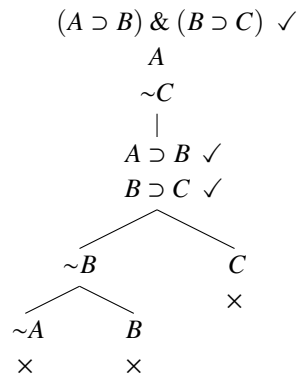
Then, because the first wff is of the form $\lceil P \supset Q \rceil$, we may apply rule (\supset) to get:

$$\begin{array}{c}
 (A \supset B) \& (B \supset C) \checkmark \\
 A \\
 \sim C \\
 | \\
 A \supset B \\
 B \supset C
 \end{array}$$

Next, we have two wffs of the form $\lceil P \supset Q \rceil$. We may start with either we wish. I will start with $\lceil B \supset C \rceil$. Applying the rule (\supset), we get:

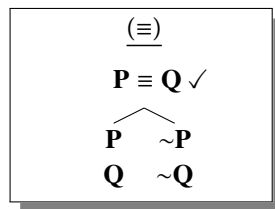


Because the far right branch has both ' $\sim C$ ' and ' C ' appearing on it, rule (\times) tells us that we must close this branch. Because there are open branches and unchecked wffs which are neither atomic nor negations of atomic formulae, we are not yet finished with the tree. We must apply the rule (\supset) to ' $A \supset B$ '. When we do so, we get:



On the left-most branch, both ' A ' and ' $\sim A$ ' appear, so that branch closes. Similarly, on the adjacent branch, both ' B ' and ' $\sim B$ ' appear, so that branch closes as well. Now, every branch has closed, so the tree closes, and we are done.

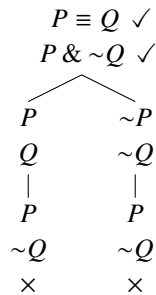
The next rule tells us what to do with wffs of the form ' $\mathbf{P} \equiv \mathbf{Q}$ '.



This rule says: if you have a wff of the form ' $\mathbf{P} \equiv \mathbf{Q}$ ' on an open branch of the tree, then, we may place a checkmark next to this wff and split every open branch on which that wff lies in two. On the left-hand-side of this split, we should write ' \mathbf{P} ' and, immediately underneath it, ' \mathbf{Q} '; on the right-hand-side, we should write ' $\sim \mathbf{P}$ ' and, immediately

underneath it, $\lceil \sim Q \rceil$.

For instance, here is a completed tree for the initial trunk consisting of the wffs ' $P \equiv Q$ ' and ' $P \& \sim Q$ ':

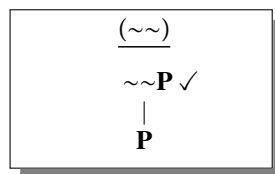


Since every branch of this tree closes, the tree closes.

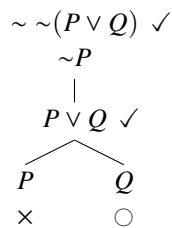
Negations

The remaining rules apply to negations—wffs whose main operators are the tilde, ' \sim '. There is not just one such rule, but rather five—one for each logical operator. In order to know which rule to apply to a negation, we must know the logical form of its *immediate subformula(e)*. A negated conjunction, for instance, has a different rule than a negation disjunction, which has a different rule than a negated conditional.

The first negation rule governs negations of negations. It tells us that, if we have a wff of the form $\lceil \sim \sim P \rceil$, then we may check this wff off and write one of the form $\lceil P \rceil$ on every open branch on which $\lceil \sim \sim P \rceil$ lies.

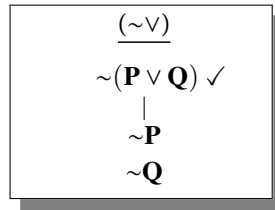


For instance, consider the tree which starts with ' $\sim \sim (P \vee Q)$ ' and ' $\sim P$ ':

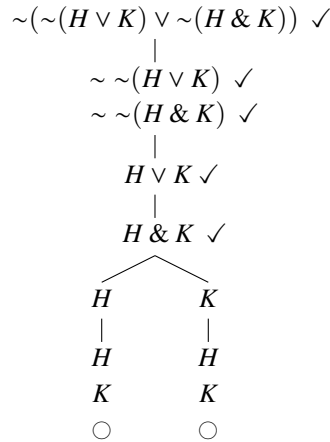


This tree has an open branch after checking off all wffs which are not atomic formulae or negations of atomic formulae. So the tree remains open.

The next negation rule governs negations of disjunctions. It says that, if you have a wff of the form $\neg(\mathbf{P} \vee \mathbf{Q})$ on an open branch, then you may place a check next to that wff; when you do so, you must write, at the bottom of every open branch on which $\neg(\mathbf{P} \vee \mathbf{Q})$ lies, write $\neg\mathbf{P}$ and, immediately beneath it, $\neg\mathbf{Q}$.

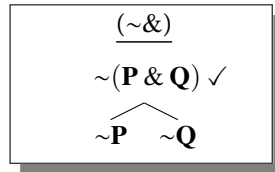


Here, for instance, is the tree beginning with the sole wff $\neg(\neg(H \vee K) \vee \neg(H \& K))$:

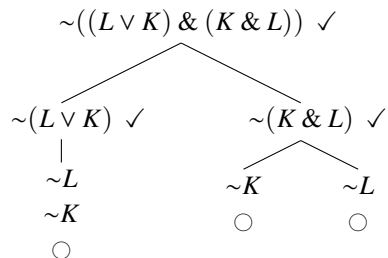


At least one branch of the tree has remained open, so the tree is open.

The next rule governs negations of conjunctions. It says that, if you have a wff of the form $\neg(\mathbf{P} \& \mathbf{Q})$ on an open branch, then you may place a check next to that wff; when you do so, you must split every open branch on which $\neg(\mathbf{P} \& \mathbf{Q})$ lies; on the left-hand-side of the split, you must write $\neg\mathbf{P}$ and on the right-hand-side of the split, you must write $\neg\mathbf{Q}$.

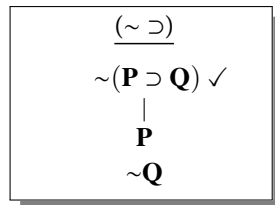


For instance, here is the tree which begins with the wff ' $\sim((L \vee K) \& (K \equiv L))$ ':

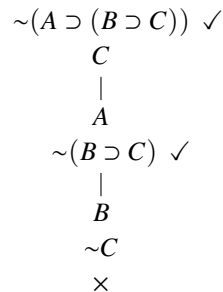


Because there is a branch of this tree which remains open, the tree is open.

The next rule governs negations of conditionals. It says that, if you have a wff of the form ' $\sim(\mathbf{P} \supset \mathbf{Q})$ ' on an open branch, then you may place a check next to that wff; when you do so, you must write, at the bottom of every open branch on which ' $\sim(\mathbf{P} \supset \mathbf{Q})$ ' lies, ' \mathbf{P} ' and, immediately beneath it, ' $\sim \mathbf{Q}$ '.

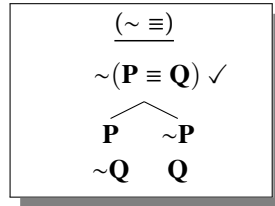


For instance, consider the tree which begins with the wffs ' $\sim(A \supset (B \supset C))$ ' and ' C ':

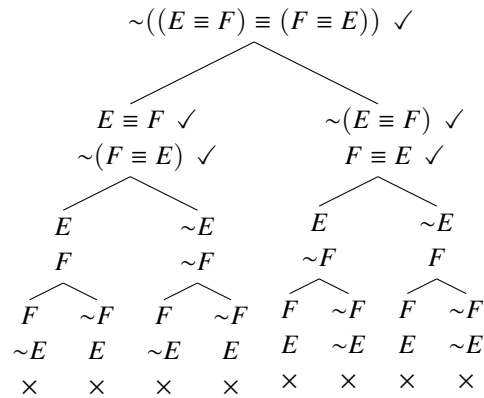


Because every branch of the tree closes, the tree closes.

The final rule governs negations of biconditionals. It says that, if you have a wff of the form $\neg(\mathbf{P} \equiv \mathbf{Q})$ on an open branch, then you may place a check next to that wff; when you do so, you must split every open branch on which $\neg(\mathbf{P} \equiv \mathbf{Q})$ lies; on the left-hand-side of the split, you must write \mathbf{P} and, immediately beneath it, $\neg\mathbf{Q}$; on the right-hand-side of the split, you must write $\neg\mathbf{P}$ and, immediately beneath it, \mathbf{Q} .



Consider, for instance, the following tree, which begins with the wff $\neg((E \equiv F) \equiv (F \equiv E))$:



Because every branch of the tree closes, the tree closes.

3.3 Summary of Rules

$$\begin{array}{c} \underline{(\vee)} \\ \mathbf{P \vee Q} \checkmark \\ \wedge \\ \mathbf{P \quad Q} \end{array}$$

$$\begin{array}{c} \underline{(\sim\vee)} \\ \sim(\mathbf{P \vee Q}) \checkmark \\ | \\ \sim\mathbf{P} \\ \sim\mathbf{Q} \end{array}$$

$$\begin{array}{c} \underline{(\&)} \\ \mathbf{P \& Q} \checkmark \\ | \\ \mathbf{P} \\ \mathbf{Q} \end{array}$$

$$\begin{array}{c} \underline{(\sim\&)} \\ \sim(\mathbf{P \& Q}) \checkmark \\ \wedge \\ \sim\mathbf{P} \quad \sim\mathbf{Q} \end{array}$$

$$\begin{array}{c} \underline{(\supset)} \\ \mathbf{P \supset Q} \checkmark \\ \wedge \\ \sim\mathbf{P} \quad \mathbf{Q} \end{array}$$

$$\begin{array}{c} \underline{(\sim\supset)} \\ \sim(\mathbf{P \supset Q}) \checkmark \\ | \\ \mathbf{P} \\ \sim\mathbf{Q} \end{array}$$

$$\begin{array}{c} \underline{(\equiv)} \\ \mathbf{P \equiv Q} \checkmark \\ \wedge \\ \mathbf{P \quad \sim P} \\ \mathbf{Q \quad \sim Q} \end{array}$$

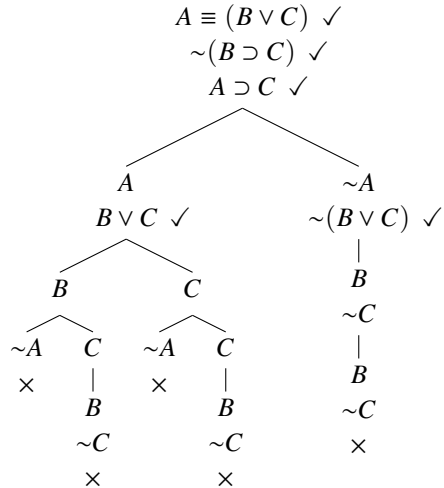
$$\begin{array}{c} \underline{(\sim\equiv)} \\ \sim(\mathbf{P \equiv Q}) \checkmark \\ \wedge \\ \mathbf{P \quad \sim P} \\ \sim\mathbf{Q \quad Q} \end{array}$$

$$\begin{array}{c} \underline{(\sim\sim)} \\ \sim\sim\mathbf{P} \checkmark \\ | \\ \mathbf{P} \end{array}$$

$$\begin{array}{c} \underline{(\times)} \\ \mathbf{P} \\ | \\ \sim\mathbf{P} \\ \times \end{array}$$

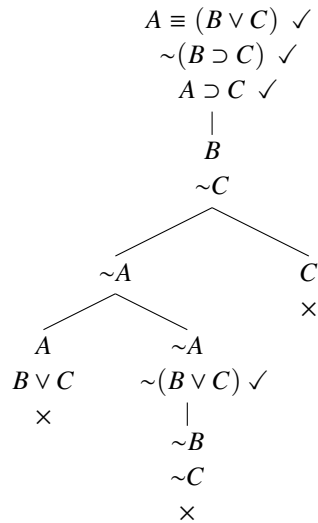
3.4 Strategies for Applying Rules

We may apply rules in any order; and, consequently, there are multiple correct trees for any given trunk. For instance, consider a tree which begins with the wffs ' $A \equiv (B \vee C)$ ', ' $\sim(B \supset C)$ ', and ' $A \supset C$ '. Here is one possible completed tree:



(In this tree, I began by first applying the rule (\equiv) to ' $A \equiv (B \vee C)$ '. I next applied the rule (\vee) to ' $B \vee C$ ' on the first branch, and the rule ($\sim \vee$) to ' $\sim(B \vee C)$ ' on the second branch. After that, I applied the rule (\supset) to ' $A \supset C$ ' to all the open branches beneath it; and finally, I applied the rule ($\sim \supset$) to ' $\sim(B \supset C)$ '.)

Here is another way of completing the tree:

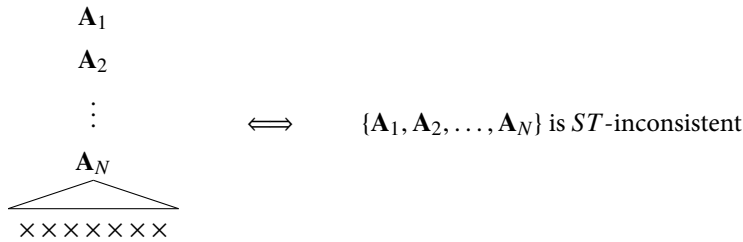


This tree is much simpler than the one we began with. In general, your choice about which order to apply the rules can end up making a difference to how long it takes a tree to close or how complicated your final tree ends up looking. Here are some rough-and-ready guidelines to follow to keep your trees as tidy as possible:

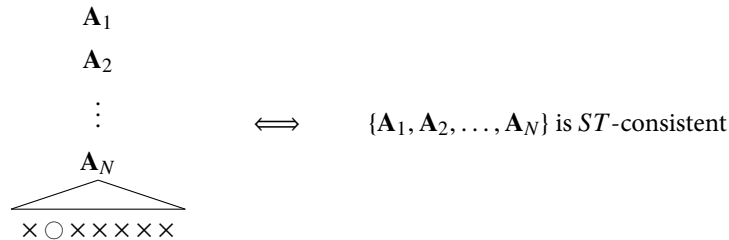
1. All else being equal, apply non-branching rules first.
2. All else being equal, apply the rules to wffs which will lead to at least some branches closing before applying the rules to wffs which will not.
3. All else being equal, work on longer wffs first.

3.5 *ST*-Consistency

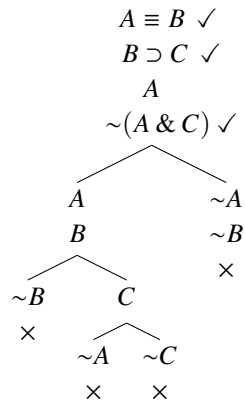
Suppose that you begin a tree with a set of wffs of *SL*, $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$, at the top, you apply the rules, and every branch of the tree closes (*i.e.*, the tree closes). Then, the set of sentences $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is *ST*-inconsistent.



Suppose, on the other hand, that you begin a tree with a set of wffs of *SL*, $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$, at the top, you apply the rules, and *not* every branch of the tree closes—at least one branch remains open (*i.e.*, the tree *doesn't* close). Then, the set of sentences $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is *ST*-consistent.

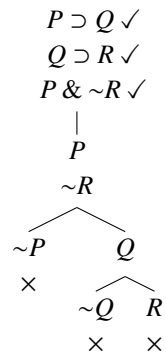


Is the set $\{A \equiv B, B \supset C, A, \sim(A \& C)\}$ *ST*-consistent?



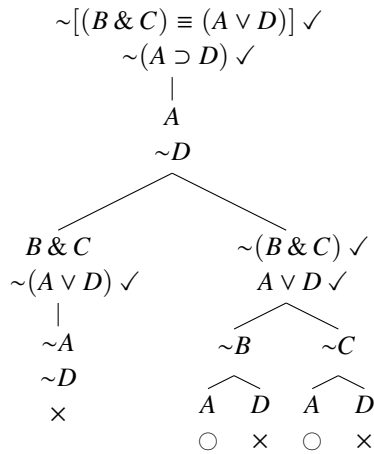
Because the tree closes, $\{A \equiv B, B \supset C, A, \sim(A \& C)\}$ is *ST*-inconsistent.

Is the set $\{P \supset Q, Q \supset R, P \& \sim R\}$ *ST*-consistent?



Because the tree closes, we know that the set $\{P \supset Q, Q \supset R, P \& \sim R\}$ is *ST*-inconsistent.

Is the set $\{\sim[(B \& C) \equiv (A \vee D)], \sim(A \supset D)\}$ *ST*-consistent?



Because the tree remains open, the set $\{\sim[(B \& C) \equiv (A \vee D)], \sim(A \supset D)\}$ is *ST*-consistent.

3.6 Reading Truth-Value Assignments off of Open Branches

It's important to note that the properties of *ST*-consistency and *ST*-inconsistency are defined very differently than the properties of *SL*-consistency and *SL*-inconsistency. The definition of *SL*-(in)consistency has to do with truth-value assignments,

SL-CONSISTENCY A set of wffs, $\{A_1, A_2, \dots, A_N\}$ is *SL*-consistent if and only if there is some truth-value assignment which makes each of $\lceil A_1 \rceil, \lceil A_2 \rceil, \dots, \lceil A_N \rceil$ true.

SL-INCONSISTENCY A set of wffs, $\{A_1, A_2, \dots, A_N\}$ is *SL*-inconsistent if and only if there is no truth-value assignment which makes each of $\lceil A_1 \rceil, \lceil A_2 \rceil, \dots, \lceil A_N \rceil$ true.

Whereas the definition of *ST*-(in)consistency has nothing to do with truth-value assignments, and everything to do with whether certain trees close or not.

ST-CONSISTENCY A set of wffs, $\{A_1, A_2, \dots, A_N\}$ is *ST*-consistent if and only if every truth tree which starts with $\lceil A_1 \rceil, \lceil A_2 \rceil, \dots, \lceil A_N \rceil$ remains open.

ST-INCONSISTENCY A set of wffs, $\{A_1, A_2, \dots, A_N\}$ is *ST*-inconsistent if and only if every truth tree which starts with $\lceil A_1 \rceil, \lceil A_2 \rceil, \dots, \lceil A_N \rceil$ closes.

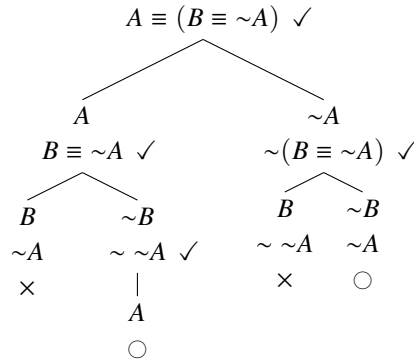
Nevertheless, there is an important relationship between *SL*-(in)consistency and *ST*-(in)consistency. While we will have to wait until later in the class to see the proof of this relationship, let me go ahead and inform you now that a set of wffs of *SL* is *SL*-consistent iff it is *ST*-consistent; and a set of wffs of *SL* is *SL*-inconsistent iff it is *ST*-inconsistent.

Fact: For any set of wffs of SL , $\{A_1, A_2, \dots, A_N\}$, that set is SL -consistent if and only if it is ST -consistent, and SL -inconsistent if and only if it is ST -inconsistent.

This fact tells us that ST -(in)consistency is a property worth paying attention to. We can use the trees to tell us something about truth-value assignments. If a tree closes, then we know that there is no truth-value assignment that makes all the wffs at the base of the tree true. If the tree remains open, then we know that there is a truth-value assignment which makes all of the wffs at the base of the tree true.

In fact, we can use truth trees to do more than this. If a tree remains open, we may use the open branches of the tree to *read off* truth-value assignments which make the wffs at the root of the tree true.

Here's an example to illustrate how we can do that: suppose that we start off with the set of wffs $\{A \equiv (B \equiv \sim A)\}$. Here is a completed tree for this set of wffs:



This tree has four branches. The first and the third close; whereas the second and the fourth remain open. Look to the atomic wffs and negated atomic wffs which show up on the second and fourth branches of the tree. On the second, we have 'A' and '~B'. On the fourth, we have '~A' and '~B'. The atomic wffs on the second branch are true iff A is true and B is false. The atomic wffs on the fourth branch are true iff both A and B are false.

Now consider the truth-table for ' $A \equiv (B \equiv \sim A)$ ':

A	B	A	\equiv	(B	\equiv	\sim	A)
T	T	T	F	T	F	F	T
T	F	T	T	F	F	F	T
F	T	F	F	T	T	T	F
F	F	F	T	F	F	T	F

The truth-table shows us that ' $A \equiv (B \equiv \sim A)$ ' is true 1) when A is true and B is false; and 2) when both A and B are false.

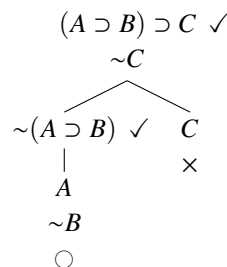
This suggests the following idea: if a tree remains open, then you may find a partial truth-value assignment which makes all of the wffs at the top of the tree true as follows: pick an open branch and look at all of the wffs on that branch which are either atomic or negations of atomic wffs. If an atomic wff ' \mathbf{P} ' appears on that branch, then let ' \mathbf{P} ' be true. If a negation of an atomic wff ' $\sim\mathbf{P}$ ' appears on that branch, then let ' \mathbf{P} ' be false. If neither ' \mathbf{P} ' nor ' $\sim\mathbf{P}$ ' appears on that branch, then you may let ' \mathbf{P} ' be either true or false—it won't matter.

To find a partial truth-value assignment which makes all the wffs at the base of an open tree true,

1. Select an open branch.
2. If an atomic wff ' \mathbf{P} ' appears on that branch, then let ' \mathbf{P} ' be true.
3. If a negation of an atomic wff ' $\sim\mathbf{P}$ ' appears on that branch, then let ' \mathbf{P} ' be false.
4. If neither ' \mathbf{P} ' nor ' $\sim\mathbf{P}$ ' appears on that branch, then you may let ' \mathbf{P} ' be either true or false.

This method will always work out. You can take my word for it for now—though that is something that we will have to prove later on in the course, as well. You may use this method to check whether you have done a tree correctly, assuming that the tree remains open. If it does, then you may plug in the partial truth-value assignment determined by one of the open branches and check to make sure that all of the wffs at the top of the tree are true on that partial truth-value assignment. (If they are not all true, then you know that you've done something wrong).

For instance, consider the following tree:



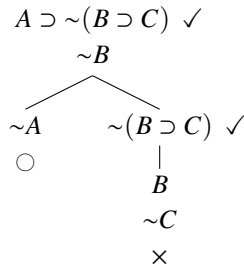
On the open branch in this tree, we have the wffs ‘ A ’, ‘ $\sim B$ ’, and ‘ $\sim C$ ’. This gives us the following partial true-value assignment:

A is true
 B is false
 C is false

Looking at this row of the truth-table we see that both wffs in $\{(A \supset B) \supset C, \sim C\}$ are true:

A	B	C		$(A \supset B)$	\supset	C		\sim	C
T	F	F		T	F	F		T	F

For another example, consider the tree which begins with the set of wffs $\{A \supset \sim(B \supset C), \sim B\}$:



‘ $\sim A$ ’ and ‘ $\sim B$ ’ appear on the only open branch of this tree. Neither ‘ C ’ nor ‘ $\sim C$ ’ appear on this branch. So we may let C be either true or false. That is: this branch suggests the following two partial truth-value assignments:

A is false	A is false
B is false	B is false
C is true	C is false

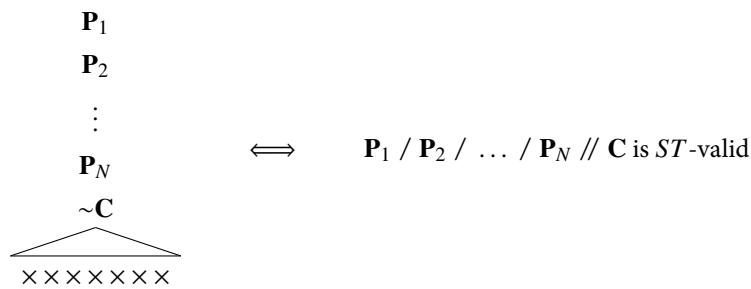
And, when we check these truth-value assignments, we see that they make both ‘ $A \supset \sim(B \supset C)$ ’ and ‘ $\sim B$ ’ true:

A	B	C		A	\supset	\sim	$(B \supset C)$		\sim	B
F	F	T		F	T	F	F	T	T	F
F	F	F		F	T	F	F	T	T	F

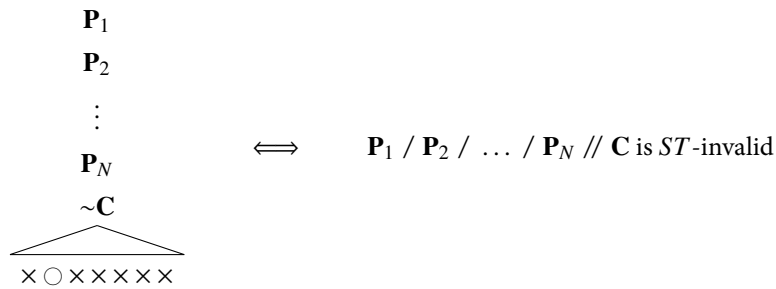
3.7 ST-Validity

We saw before that an *SL*-argument $\mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C}$ is *SL*-valid if and only if the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim\mathbf{C}\}$ is *SL*-inconsistent. Since the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim\mathbf{C}\}$ is *SL*-inconsistent if and only if it is *ST*-inconsistent, we may use truth-trees to test arguments for *SL*-validity.

Let's define a new notion: *ST*-validity. An *SL* argument is *ST*-valid if and only if the tree beginning with the wffs $\lceil \mathbf{P}_1 \rceil$, $\lceil \mathbf{P}_2 \rceil$, ..., $\lceil \mathbf{P}_N \rceil$, and $\lceil \sim\mathbf{C} \rceil$ closes.



Similarly, we will say that an *SL*-argument is *ST*-invalid if and only if the tree beginning with the wffs $\lceil \mathbf{P}_1 \rceil$, $\lceil \mathbf{P}_2 \rceil$, ..., $\lceil \mathbf{P}_N \rceil$, and $\lceil \sim\mathbf{C} \rceil$ remains open.



It's important to distinguish between *SL*-(in)validity and *ST*-(in)validity. The former is defined in terms of truth-value assignments; whereas the latter is defined in terms of the truth-trees. As it turns out, these two notions are deeply connected. Towards the end of the course, we will prove that an argument is *SL*-valid if and only if it is *ST*-valid, and that it is *SL*-invalid if and only if it is *ST*-invalid.

Fact: For any *SL*-argument $\mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C}$, that argument is *SL*-valid if and only if it is *ST*-valid, and *SL*-invalid if and only if it is *ST*-invalid.

However, this is something that we will have to prove. It is not obviously or straightforwardly true.

Consider the following *SL*-argument:

$$L \supset \sim M / N \supset M / L \supset N // \sim L$$

To test this argument for *ST*-validity, we begin by placing its premises and the negation of its conclusion at the top of a tree, as follows:

$$\begin{array}{c} L \supset \sim M \\ N \supset M \\ L \supset N \\ \sim \sim L \end{array}$$

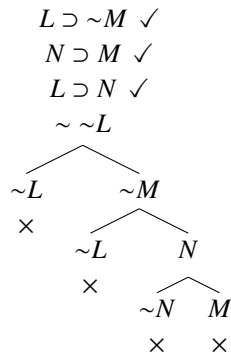
We may then apply the rule for \supset to the first premise, to get:

$$\begin{array}{c} L \supset \sim M \checkmark \\ N \supset M \\ L \supset N \\ \sim \sim L \\ \wedge \\ \sim L \quad \sim M \\ \times \end{array}$$

Applying the rule for \supset to the third premise, we get:

$$\begin{array}{c} L \supset \sim M \checkmark \\ N \supset M \\ L \supset N \checkmark \\ \sim \sim L \\ \wedge \\ \sim L \quad \sim M \\ \times \quad \wedge \\ \quad \sim L \quad N \\ \quad \times \end{array}$$

Finally, applying the rule for \supset to the second premise, we get:



Because the tree closes, we know that $L \supset \sim M / N \supset M / L \supset N // \sim L$ is *ST*-valid.

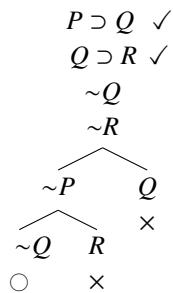
Next, consider the *SL*-argument

$$P \supset Q / Q \supset R / \sim Q // R$$

To see whether this argument is *ST*-valid, we place its premises and the negation of its conclusion at the top of a tree, as follows:

$$\begin{array}{c}
 P \supset Q \\
 Q \supset R \\
 \sim Q \\
 \sim R
 \end{array}$$

Completing the tree, we arrive at:



Since the tree remains open, we know that the *SL*-argument $P \supset Q / Q \supset R / \sim Q // R$ is *ST*-invalid.

Moreover, we may use the tree to read off a partial truth-value assignment which makes the premises of the argument all true yet makes its conclusion false. We proceed exactly as before. On the only open branch of the tree, we have the wffs ' $\sim P$ ', ' $\sim Q$ ', and ' $\sim R$ '. This

gives us the partial truth-value assignment

P is false
Q is false
R is false

This partial truth-value assignment is what we will call an *ST-counterexample* to the validity of the argument $P \supset Q / Q \supset R / \sim Q // R$.

If we look at the corresponding row of the truth-table, we find that it is additionally an *SL-counterexample* to the *SL*-validity of the argument:

<i>P</i>	<i>Q</i>	<i>R</i>	<i>P</i>	\supset	<i>Q</i>	<i>Q</i>	\supset	<i>R</i>	\sim	<i>Q</i>	\parallel	<i>R</i>
F	F	F	F	T	F	F	T	F	T	F		F

ST-counterexample An *ST-COUNTEREXAMPLE* to the *ST*-validity of an *SL*-argument is a partial truth-value assignment read off of an open branch of the truth tree containing the argument’s premises and the negation of the argument’s conclusion at its root (*i.e.*, the partial truth-value assignment that we get when we follow the procedure from §3.6 above).

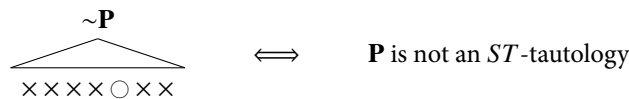
3.8 *ST*-Tautologies, *ST*-Contradictions, and *ST*-Contingencies

3.8.1 *ST*-Tautologies

A wff of *SL*, $\ulcorner \mathbf{P} \urcorner$, is an *ST*-tautology if and only if a tree with $\ulcorner \sim \mathbf{P} \urcorner$ at its root closes.



Correlatively, a wff of *SL* is *not* an *ST*-tautology if and only if a tree with $\ulcorner \sim \mathbf{P} \urcorner$ at its root remains open.



Here, again, it is important to keep the notion of an *ST*-tautology separate from the notion of an *SL*-tautology. The definition of an *SL*-tautology has to do with truth-value

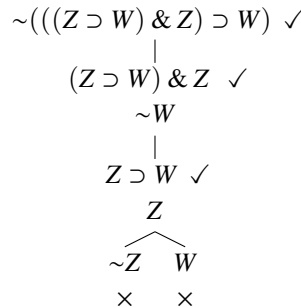
assignments ($\lceil \mathbf{P} \rceil$ is an *SL*-tautology iff $\lceil \mathbf{P} \rceil$ is true on every truth-value assignment), whereas the definition of an *ST*-tautology has nothing to do with truth-value assignments and everything to do with whether, when we apply the tree rules, our tree closes.

Now, it turns out that these two notions line up perfectly; that is, it turns out that:

Fact: A wff of *SL* is an *SL*-tautology if and only if it is an *ST*-tautology.

That is: it turns out that a wff of *SL* is true on every truth-value assignment if and only if the tree which starts with the negation of that wff at its root closes. However, this is something that we will have to prove; and not something that we may simply take for granted.

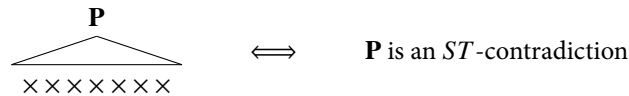
Consider, for instance, the wff of *SL* $((Z \supset W) \& Z) \supset W$. When we put the negation of this wff at the root of a tree and apply the rules, we get:



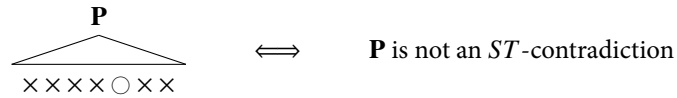
Since every branch of the tree closes, the tree closes; and thus, the wff $((Z \supset W) \& Z) \supset W$ is an *ST*-tautology.

3.8.2 *ST*-Contradictions

A wff of *SL*, $\lceil \mathbf{P} \rceil$, is an *ST*-contradiction if and only if a tree with $\lceil \mathbf{P} \rceil$ at its root closes.



Correlatively, a wff of *SL* is *not* an *ST*-tautology if and only if a tree with $\lceil \sim \mathbf{P} \rceil$ at its root remains open.

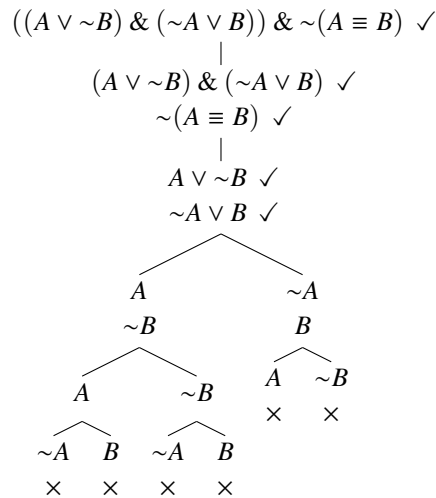


Again, it turns out that the notion of an *SL*-contradiction lines up perfectly with the notion of an *ST*-contradiction; that is, it turns out that:

Fact: A wff of *SL* is an *SL*-contradiction if and only if it is an *ST*-contradiction.

That is: it turns out that a wff of *SL* is false on every truth-value assignment if and only if the tree which starts with that wff at its root closes. We will prove this later in the course.

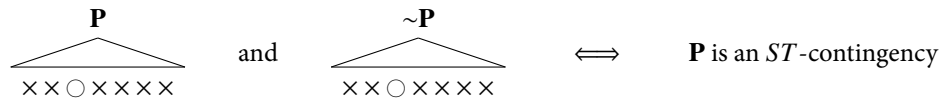
Consider the wff $\lceil ((A \vee \sim B) \& (\sim A \vee B)) \& \sim(A \equiv B) \rceil$. When we construct the tree with this wff at its root, we get:



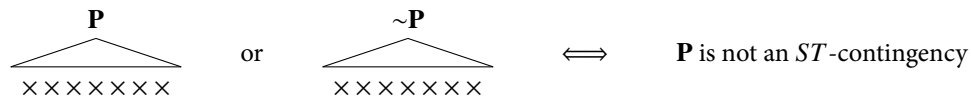
Since the tree with $\lceil ((A \vee \sim B) \& (\sim A \vee B)) \& \sim(A \equiv B) \rceil$ at its root closes, that wff is an *ST*-contradiction.

3.8.3 ST-Contingencies

A wff of SL , $\lceil \mathbf{P} \rceil$, is an ST -contingency if and only if *both* a tree with $\lceil \sim \mathbf{P} \rceil$ at its root closes *and* a tree with $\lceil \mathbf{P} \rceil$ at its root closes.



Correlatively, a wff of SL is *not* an ST -contingency if and only if *either* a tree with $\lceil \sim \mathbf{P} \rceil$ at its root remains open *or* a tree with $\lceil \mathbf{P} \rceil$ at its root remains open.

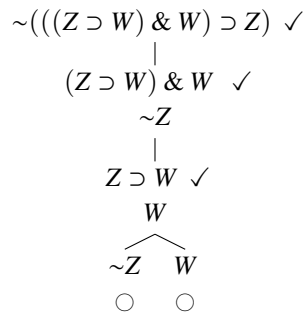


Again, it turns out that the notion of an SL -contingency lines up perfectly with the notion of an ST -contingency; that is, it turns out that:

Fact: A wff of SL is an SL -contingency if and only if it is an ST -contingency.

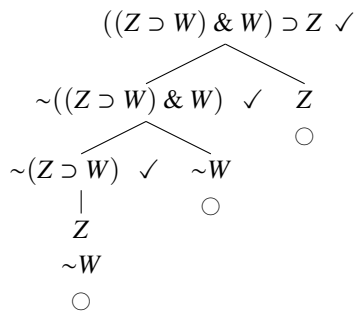
That is: it turns out that a wff of SL is true on some truth-value assignments and false on other truth-value assignments if and only if both the tree which starts with that wff at its root remains open and the tree which starts with the negation of that wff at its root remains open. We will prove this later in the course.

Consider the wff $\lceil ((Z \supset W) \& W) \supset Z \rceil$. When we put the negation of this wff at the root of a tree and apply the rules, we get:



Because the tree remains open, this tells us that the wff $\lceil ((Z \supset W) \& W) \supset Z \rceil$ is not an ST -tautology.

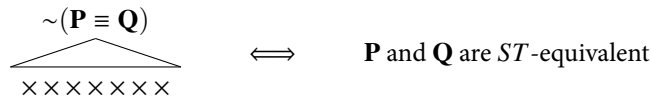
When we put $\lceil ((Z \supset W) \& W) \supset Z \rceil$ at the root of a tree and apply the tree rules, we get:



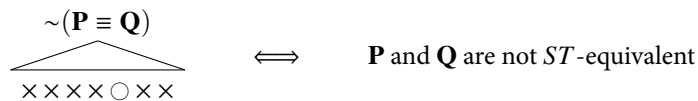
Because the tree remains open, this tells us that ‘ $((Z \supset W) \& W) \supset Z$ ’ is not an *ST*-contradiction. Thus, ‘ $((Z \supset W) \& W) \supset Z$ ’ is an *ST*-contingency.

3.9 ST-Equivalence

A pair of wffs of *SL*, ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’, are *ST*-equivalent if and only if a tree with ‘ $\sim(\mathbf{P} \equiv \mathbf{Q})$ ’ at its root closes.



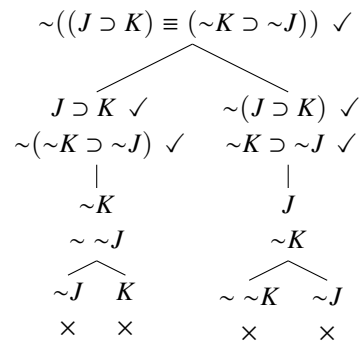
Correlatively, a pair of wffs of *SL*, ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’, are *not ST*-equivalent if and only if a tree with ‘ $\sim(\mathbf{P} \equiv \mathbf{Q})$ ’ at its root remains open.



As before, it is important to keep the notion of *ST*-equivalence distinct from the notion of *SL*-equivalence. Two wffs of *SL* are *SL*-equivalent if and only if there is no truth-value assignment on which they have different truth-values. Two wffs of *SL* are *ST*-equivalent iff a certain tree formed by correct application of the rules for trees ends up with \times ’s at the end of all of its branches. These are very different kinds of properties. However, it turns out that the two of them line up perfectly. That is, it turns out that:

Fact: Two wffs of *SL* are *SL*-equivalent if and only if they are *ST*-equivalent.

Consider the pair of wffs ‘ $J \supset K$ ’ and ‘ $\sim K \supset \sim J$ ’. To see that these two wffs are *ST*-equivalent, we can construct the tree which begins with ‘ $\sim((J \supset K) \equiv (\sim K \supset \sim J))$ ’:



Since this tree closes, we know that ' $J \supset K$ ' and ' $\sim K \supset \sim J$ ' are *ST*-equivalent.

Chapter 4

Predicate Logic

4.1 The Language PL

Before getting into the nitty-gritting, some preliminary orientation: we're going to use capital letters to denote *properties* that a thing might or might not have and *relations* things might or might not bear to one another, and we're going to use lowercase letters to denote the things that may or may not have those properties or may and may not bear those relations to one another. So, for instance, we could use the capital letters T , L , and K to represent the following properties and relations:

$Tx = x$ was tall

$Lxy = x$ loved y

$Kxy = x$ killed y

and we could use l , b , c , and p to represent the following individuals:

$l =$ Abraham Lincoln

$b =$ John Wilkes Booth

$c =$ Caesar

$p =$ Pompey

If we put the lowercase letters representing individuals in the place of ‘ x ’ and ‘ y ’ above, we get statements like the following:

$$\begin{aligned} Tl &= \text{Abraham Lincoln was tall} \\ Kbl &= \text{John Wilkes Booth killed Abraham Lincoln} \\ Lcp &= \text{Caesar loved Pompey} \end{aligned}$$

We can treat these statements the same way that we treated the statement letters of SL —they can be the negands of negations, the antecedents of conditionals, the disjuncts of disjunctions, and so on and so forth.

$$\begin{aligned} \sim Lbl &= \text{John Wilkes Booth didn't love Abraham Lincoln} \\ Kcp \supset \sim Lcp &= \text{If Caesar killed Pompey, then he didn't love him} \\ Tc \vee Tb &= \text{Either Caesar or John Wilkes Booth is tall} \end{aligned}$$

We’re also going to be able to translate claims like ‘everyone loves someone’ and ‘no one loves anyone who killed them.’ They will be translated like so:

$$\begin{aligned} (\forall x)(\exists y)Lxy &= \text{Everyone loves someone} \\ \sim(\exists x)(\exists y)(Kyx \ \& \ Lxy) &= \text{No one loves anyone who killed them} \end{aligned}$$

But in order to understand that, we’ll have to get into the nitty-gritty.

4.1.1 The Syntax of PL

In this section, I’m going to tell you what the vocabulary of PL is and I’m going to tell you which expressions of PL are grammatical—which are *well-formed*—just as we did for SL .

Vocabulary

The vocabulary of PL includes the following symbols:

1. for each $n \geq 0$, an infinite number of n -place predicates (any capital letter, along

with a superscript n —perhaps with subscripts)

$$\begin{array}{cccccccc}
 A^1 & B^1 & \dots & Z^1 & A_1^1 & \dots & Z_1^1 & A_2^1 & \dots \\
 A^2 & B^2 & \dots & Z^2 & A_1^2 & \dots & Z_1^2 & A_2^2 & \dots \\
 \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \dots \\
 A^n & B^n & \dots & Z^n & A_1^n & \dots & Z_1^n & A_2^n & \dots \\
 \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \dots
 \end{array}$$

2. An infinite number of *constants* (any lowercase letter between a and v —perhaps with subscripts)

$$a, b, c, \dots, u, v, a_1, b_1, \dots, v_1, a_2, b_2, \dots$$

3. An infinite number of *variables* (lowercase $w, x, y,$ or z —perhaps with subscripts)

$$w, x, y, z, w_1, x_1, y_1, z_1, w_2, x_2, \dots$$

4. Logical operators

$$\sim, \vee, \&, \supset, \equiv, \exists, \forall$$

5. parentheses

$$(,)$$

Nothing else is included in the vocabulary of *PL*.

Terminology: Let's call both constants and variables *terms*. That is, both ' a ' and ' x ' are *terms* of *PL*.

Grammar

Any sequence of the symbols in the vocabulary of *PL* is a *formula* of *PL*. For instance, all of the following are formulae of *PL*:

$$\begin{array}{l}
 V^{2800}x \sim ((\supset \supset) anv \\
 P^1Q^2R^3S^4T^5 \sim \sim \\
 ((\forall x)F^3xab \supset \sim(\exists y)P^4ynst) \\
 N^{54}xy\vee \sim \sim(\exists x)B^2x
 \end{array}$$

However, only one—the third—is a *well-formed formula* (or ‘wff’) of PL . We specify what it is for a string of symbols from the vocabulary of PL to be a wff of PL with the following rules.

- \mathcal{F}) If $\lceil \mathcal{F}^n \rceil$ is an n -place predicate and $\lceil \mathbf{t}_1 \rceil, \lceil \mathbf{t}_2 \rceil, \dots, \lceil \mathbf{t}_n \rceil$ are n terms, then $\lceil \mathcal{F}^n \mathbf{t}_1 \mathbf{t}_2 \dots \mathbf{t}_n \rceil$ is a wff.
- \sim) If $\lceil \mathbf{P} \rceil$ is a wff, then $\lceil \sim \mathbf{P} \rceil$ is a wff.
- $\&$) If $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are wffs, then $\lceil (\mathbf{P} \& \mathbf{Q}) \rceil$ is a wff.
- \vee) If $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are wffs, then $\lceil (\mathbf{P} \vee \mathbf{Q}) \rceil$ is a wff.
- \supset) If $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are wffs, then $\lceil (\mathbf{P} \supset \mathbf{Q}) \rceil$ is a wff.
- \equiv) If $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are wffs, then $\lceil (\mathbf{P} \equiv \mathbf{Q}) \rceil$ is a wff.
- \forall) If $\lceil \mathbf{P} \rceil$ is a wff and $\lceil \mathbf{x} \rceil$ is a variable, then $\lceil (\forall \mathbf{x}) \mathbf{P} \rceil$ is a wff.
- \exists) If $\lceil \mathbf{P} \rceil$ is a wff and $\lceil \mathbf{x} \rceil$ is a variable, then $\lceil (\exists \mathbf{x}) \mathbf{P} \rceil$ is a wff.
- Nothing else is a wff.

Note: none of ‘ \mathcal{F} ’, ‘ \mathbf{a} ’, ‘ \mathbf{P} ’, and ‘ \mathbf{Q} ’ appear in the vocabulary of PL . They are not *themselves* wffs of PL . Rather, we are using them here as VARIABLES ranging over the formulae of PL . In PL , we used lowercase letters for this purpose. However, in PL , lowercase letters are *terms* of the language, so we must use other symbols for the variables ranging over the formulae of PL . We have chosen to use boldface and script letters for this purpose. Capital script letters are variables ranging over the *predicates* of PL ; boldface capital letters are variables ranging over *wffs* of PL ; and boldface lowercase letters are variables ranging over the *terms* of PL .

All and only the strings of symbols that can be constructed by repeated application of the rules above are well-formed formulae. For instance, if we wanted to show that ‘ $((\forall y)F^1y \supset \sim(\exists x)(\exists z)G^2zx)$ ’ is a wff of PL , we could walk through the following steps to build it up:

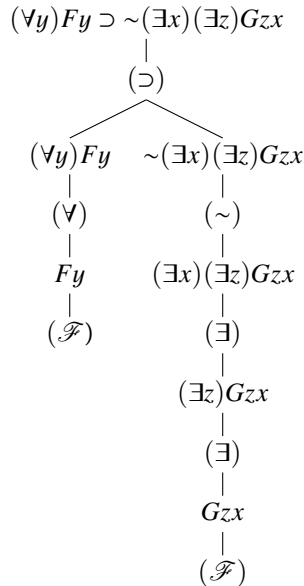
- a) ' F^1y ' is a wff [from (\mathcal{F})]
 b) So, ' $(\forall y)F^1y$ ' is a wff [from (a) and (\forall)]
 c) ' G^2zx ' is a wff [from (\mathcal{F})]
 d) So, ' $(\exists z)G^2zx$ ' is a wff [from (c) and (\exists)]
 e) So, ' $(\exists x)(\exists z)G^2zx$ ' is a wff [from (d) and (\exists)]
 f) So, ' $\sim(\exists x)(\exists z)G^2zx$ ' is a wff [from (e) and (\sim)]
 g) So, ' $((\forall y)F^1y \supset \sim(\exists x)(\exists z)G^2zx)$ ' is a wff [from (b), (f), and (\supset)]

As before, we will adopt the convention of dropping the outermost parentheses in a wff of PL . We will *additionally* adopt the convention of dropping the superscripts on the predicates of PL . So, abiding by our informal conventions, we would write the wff of PL ' $((\forall y)F^1y \supset \sim(\exists x)(\exists z)G^2zx)$ ' as:

$$(\forall y)Fy \supset \sim(\exists x)(\exists z)Gzx$$

I'll adopt these conventions from here on out.

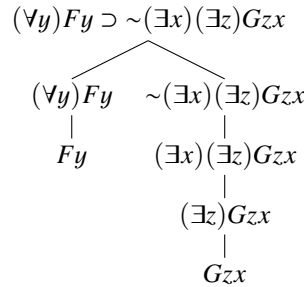
We could, just as before, use SYNTAX TREES to represent the way that a wff of PL is built up according to the rules for wffs given above. For instance, we could notate the proof given above as follows:



The left-hand branch of this syntax tree tells us that ' Fy ' is a wff, by rule (\mathcal{F}); and that, therefore, ' $(\forall y)Fy$ ' is a wff, by rule (\forall). The right-hand branch tells us that ' Gzx ' is a wff, by rule (\mathcal{F}); and that, therefore, ' $(\exists z)Gzx$ ' is a wff, by rule (\exists). Thus, ' $(\exists x)(\exists z)Gzx$ ' is a wff, by rule (\exists) again; and, finally, that, therefore, ' $\sim(\exists x)(\exists z)Gzx$ ' is a wff, by rule (\sim). Putting together what we have from the left-hand branch and the right-hand branch, we can conclude that ' $(\forall y)Fy \supset \sim(\exists x)(\exists z)Gzx$ ' is a wff, by rule (\supset).

That is to say: the syntax tree tells us exactly what the proof above tells us. It tells us how we may show that ' $(\forall y)Fy \supset \sim(\exists x)(\exists z)Gzx$ ' is a wff of PL by building it up out of its components, according to the rules for wffs for PL .

If we want a simpler way of notating a syntax tree like this, then we may simply remove the justifications (recognizing that they are clear from the context of what lies above each wff on the syntax tree), and write it out as follows:



Free and Bound Variables

Our rules for wffs count ' Fx ' and ' Ayc ' as well-formed formulae. However, the variables that appear in these wffs are **FREE**. On the other hand, the variables appearing in ' $(\forall x)(\forall y)Fxy$ ' are **BOUND**. In ' $(\forall x)Px \supset Qx$ ', the first occurrence of the variable ' x ' is bound, whereas the second occurrence is free.

To make these ideas precise, let's introduce the idea of a **QUANTIFIER**. For any variable ' \mathbf{x} ', both ' $(\forall \mathbf{x})$ ' and ' $(\exists \mathbf{x})$ ' are *quantifiers*. We call ' $(\forall \mathbf{x})$ ' the **UNIVERSAL QUANTIFIER**, and we call ' $(\exists \mathbf{x})$ ' the **EXISTENTIAL QUANTIFIER**. These quantifiers are logical operators. They can be the **MAIN OPERATOR** of a wff of PL or they can be the main operator of a wff's subformulae. Each quantifier has one and only one associated *variable*. For instance, the variable associated with the quantifier ' $(\forall z)$ ' is ' z '. The variable associated with the quantifier ' $(\exists y)$ ' is ' y '.

As before, we can define the **MAIN OPERATOR** of a wff of PL to be the logical operator whose associated rule is *last* appealed to when building the wff up according to the rules given above. So, for instance, the main operator of ' $(\forall y)Fy \supset \sim(\exists x)(\exists z)Gzx$ ' is the

horseshoe ‘ \supset ’. The main operator of ‘ $(\forall x)Fx$ ’, on the other hand, whose syntax tree is shown below, is the universal quantifier ‘ $(\forall x)$ ’:

$$\begin{array}{c} (\forall x)Fx \\ | \\ Fx \end{array}$$

Similarly, the main operator of ‘ $(\exists y)(Fy \ \& \ Ga)$ ’ is ‘ $(\exists y)$ ’:

$$\begin{array}{c} (\exists y)(Fy \ \& \ Ga) \\ | \\ Fy \ \& \ Ga \\ \wedge \\ Fy \quad Ga \end{array}$$

We can define **SUBFORMULA** in the same way that we defined it before: ‘ $\ulcorner P \urcorner$ ’ is a subformula of ‘ $\ulcorner Q \urcorner$ ’ if and only if ‘ $\ulcorner P \urcorner$ ’ must show up on a line during the proof that ‘ $\ulcorner Q \urcorner$ ’ is a wff of *PL*. In terms of the syntax trees: ‘ $\ulcorner P \urcorner$ ’ is a subformula of ‘ $\ulcorner Q \urcorner$ ’ if and only if ‘ $\ulcorner P \urcorner$ ’ lies somewhere on ‘ $\ulcorner Q \urcorner$ ’s syntax tree.

Similarly, we can define **IMMEDIATE SUBFORMULA** in precisely the same way as before: ‘ $\ulcorner P \urcorner$ ’ is an immediate subformula of ‘ $\ulcorner Q \urcorner$ ’ iff a line asserting that ‘ $\ulcorner P \urcorner$ ’ is a wff must be appealed to in the final line of a proof showing that ‘ $\ulcorner Q \urcorner$ ’ is a wff, according to the rules for wffs given above. In terms of the syntax trees: ‘ $\ulcorner P \urcorner$ ’ is an immediate subformula of ‘ $\ulcorner Q \urcorner$ ’ iff ‘ $\ulcorner P \urcorner$ ’ lies immediately below ‘ $\ulcorner Q \urcorner$ ’ on the syntax tree. Then, the immediate subformula of ‘ $(\forall x)Fx$ ’ is ‘ Fx ’, and the immediate subformula of ‘ $(\exists y)(Fy \ \& \ Ga)$ ’ is ‘ $Fy \ \& \ Ga$ ’.

The **SCOPE** of a quantifier appearing in a wff of *PL*, ‘ $\ulcorner P \urcorner$ ’, is the immediate subformula of the subformula of ‘ $\ulcorner P \urcorner$ ’ for which that quantifier is the main operator.

The scope of a quantifier— $(\forall x)$ or $(\exists x)$ —is the immediate subformula of the wff for which that quantifier is the main operator.

So, for instance, in the wff $(\exists y)Lyy \supset (\exists x)(\exists y)Lxy$, whose syntax tree is shown below,

$$\begin{array}{c} (\exists y)Lyy \supset (\exists x)(\exists y)Lxy \\ \wedge \\ (\exists y)Lyy \quad (\exists x)(\exists y)Lxy \\ | \qquad \qquad | \\ Lyy \qquad \qquad (\exists y)Lxy \\ \qquad \qquad \qquad | \\ \qquad \qquad \qquad Lxy \end{array}$$

The scope of the very first existential quantifier ‘ $(\exists y)$ ’ is the formula ‘ Lyy ’. The scope of

the second existential quantifier ' $(\exists x)$ ' is ' $(\exists y)Lxy$ '. And the scope of the final existential quantifier ' $(\exists y)$ ' is ' Lxy '.

Now, we can define the notions of a FREE and a BOUND variable.

A variable x in a wff of PL is BOUND if and only if it occurs within the scope of a quantifier, $(\forall x)$ or $(\exists x)$, whose associated variable is x .

A variable x in a wff of PL is FREE if and only if it does not occur within the scope of a quantifier, $(\forall x)$ or $(\exists x)$, whose associated variable is x .

For instance, in the wff

$$(\forall x)(\forall y)Fy \supset (\exists z)Gzx$$

The final occurrence of ' x ' is *free*. Even though there is a universal quantifier ' $(\forall x)$ ' in the wff, the final ' x ' does not occur within the scope of this universal quantifier, so it is not bound by it.

We can similarly define the notion of what it is for a quantifier to BIND a variable.

In a wff of PL , a quantifier $(\forall x)$ or $(\exists x)$ BINDS a variable x if and only if x occurs *free* within that quantifier's scope.

This means that a variable can only be bound by a single quantifier. So, for instance, in the following wff of PL ,

$$(\exists x)(\forall x)Fx$$

The variable ' x ' is bound by the *universal* quantifier ' $(\forall x)$ '. It is *not* bound by the existential quantifier ' $(\exists x)$ '.

Note: variable symbols (w, x, y, z) are only free or bound when they occur after a predicate letter. The variable symbols that appear *within the quantifiers themselves* are neither free nor bound. So, for instance, in the wff ' $(\forall x)(\forall x)Lxx$ ', the symbol ' x ' appearing in the second (innermost) universal quantifier is *not* bound by the first (outermost) universal quantifier. The only occurrences of the symbol ' x ' which are either free or bound are the final two, after ' L ', and both of them are bound by the second (innermost) universal quantifier.

Important Syntactic Features in PL

Parentheses will serve an important rule in distinguishing wffs of PL , just as they played an important role in distinguishing the wffs of PL . Thus, $(\forall x)((\exists y)Lxy \supset Gx)$ is a different wff than $(\forall x)(\exists y)Lxy \supset Gx$; for they have different syntax trees, as shown below:



And, in fact, one of these wffs can be true while the other is false (that is to say, they *mean* different things). So it will be very important in PL to keep track of your parentheses.

Similarly, **the order of the terms following the predicates in PL matter**. ' Lab ' is a different wff than ' Lba '; similarly, $(\forall x)(\exists y)Lxy$ is a different wff than $(\forall x)(\exists y)Lyx$. For they have different syntax trees,



(since they have different wffs on each line). Moreover, this difference is also one that will end up making a difference to the *meaning* of the wffs of PL . Again, if $Lxy = x$ loves y , and we're considering only people, then we'll end up seeing that $(\forall x)(\exists y)Lxy$ says that everybody loves somebody else; whereas $(\forall x)(\exists y)Lyx$ says that everybody is loved *by* somebody else. And these two mean very different things—we'll end up seeing that one could be true while the other is false.

Moreover, **the order of quantifiers plays an important role in distinguishing the wffs of PL** . For instance, $(\exists x)(\forall y)Lxy$ is a different wff of PL than $(\forall y)(\exists x)Lxy$. For these wffs of PL have different syntax trees, as shown below:



And, again, this is a difference that will end up making a difference. If $Lxy = x$ loves y , and we're considering only people, then we'll end up seeing—once we've given the semantics for PL , below—that ' $(\exists x)(\forall y)Lxy$ ' says that somebody loves everybody; whereas ' $(\forall y)(\exists x)Lxy$ ' says that everybody is loved by somebody. And these two mean very different things. So we must be careful to pay attention to the order of the quantifiers.

4.1.2 Semantics for PL

In SL , we defined the semantics for the language in terms of TRUTH-VALUE ASSIGNMENTS. A truth-value assignment, recall, was just an assignment of truth-value (either true or false), to all of the *statement letters* of SL .

A TRUTH-VALUE ASSIGNMENT is an assignment of truth-value—either true or false—to every statement letter of SL .

We then gave definitions for \sim , \vee , $\&$, \supset , and \equiv that allowed us to say, for any given wff of SL , whether it was true or false on that truth-value assignment. Since this allowed us to understand the circumstances under which the wffs of SL were true or false, this provided us with the *meaning* of the wffs of SL .

However, we saw that, since there were so many statement letters of SL , specifying a truth-value assignment was prohibitively difficult. So, instead, we realized that we could look at a PARTIAL TRUTH-VALUE ASSIGNMENT. Where, recall,

A PARTIAL TRUTH-VALUE ASSIGNMENT assigns a truth-value—either true or false—to each statement letter in some set of statement letters.

Each row of a truth-table represented a partial truth-value assignment. The definitions we gave of \sim , $\&$, \vee , \supset , and \equiv then allowed us to work out the truth-value of a given wff of SL in every partial truth-value assignment (that is, in every row of the truth-table).

We're going to do exactly the same thing with PL . However, rather than dealing with TRUTH-VALUE ASSIGNMENTS, we're going to deal with PL -INTERPRETATIONS.

A PL -INTERPRETATION, \mathcal{I} , provides

1. A specification of which things fall in the *domain*, \mathcal{D} , of the interpretation.
2. For every variable of PL , a specification of which thing in the domain \mathcal{D} it represents.
3. For every constant of PL , a specification of which thing in the domain \mathcal{D} it represents.
4. For every predicate of PL , a specification of the property or relation it represents.

Because a PL interpretation requires us to say of every term in the language which thing in the domain that term denotes—and because it requires us to say of every predicate in the language which property or relation it denotes—and because there are an infinite number of terms and predicates in our language, specifying a full PL -interpretation is just as difficult as specifying a full truth-value assignment. Therefore, just as we introduced the idea of a PARTIAL TRUTH-VALUE ASSIGNMENT (which were just the rows of the truth-tables in SL), we will also introduce the idea of a PARTIAL PL -INTERPRETATION.

Given a wff, set of wffs, or argument of PL , a PARTIAL PL -INTERPRETATION, \mathcal{I}^P provides:

1. A specification of which things fall in the *domain*, \mathcal{D} , of the partial interpretation.
2. For the *free* variables appearing in the wff, set of wffs, or argument of PL , a specification of which thing in the domain \mathcal{D} they represent.
3. For the constants appearing in the wff, set of wffs, or argument of PL , a specification of which thing in the domain \mathcal{D} they represent.
4. For the predicates appearing in the wff, set of wffs, or argument of PL , a specification of the property or relation they represent.

For instance, suppose that we have the following wff of PL ,

$$\sim(\forall y)Lya \supset \sim Ha$$

Here is a *partial interpretation* of this wff:

$$\mathcal{I}^P = \left\{ \begin{array}{l} \mathcal{D} = \{ \text{Adam, Betsy, Carol} \} \\ a = \text{Adam} \\ Lxy = \text{x loves y} \\ Hx = \text{x is happy} \end{array} \right.$$

We specified the domain, \mathcal{D} . Since all of the variables are bound, we do not need to say which thing they refer to. There is only one constant in the wff, ‘ a ’, and we said what that constant referred to—Adam. And there are just two predicates in this wff: a 1-place predicate ‘ H ’, and a 2-place predicate in the wff, ‘ L ’. And we said that Hx referred to the property ‘ x is happy’ and that Lxy referred to the relation ‘ x loves y ’. So we’ve provided a partial PL interpretation for this wff.

4.1.3 More on PL -Interpretations of Predicates

Above, I interpreted H by just saying that Hx means that x is happy and I interpreted L by just saying that Lxy means that x loves y . This is fine, as far as it goes, except that you might not know whether Adam is happy, or whether Adam loves Betsy, or whether Betsy loves Carol. There is another way of specifying a the meaning of a predicate of PL that will tell us precisely which objects in the domain have the relevant property or bear the relevant relation to one another.

Suppose that only Adam and Betsy are happy. Then, we could specify the meaning of H , relative to our domain, by just telling you that it applies to Adam and Betsy. Here’s a way to do that: we just give, as our interpretation of H , the set containing Adam and Betsy:

$$H = \{ \text{Adam, Betsy} \}$$

And we can do precisely the same thing with the 2-place predicate L . Suppose that both Adam and Betsy love Carol, but that nobody loves anybody else. Then, we can specify the meaning of L , relative to our domain, by just telling you that it relates Adam to Carol and that it relates Betsy to Carol. Here is a way to do that: we just give, as our interpretation of L , the set containing the ordered pairs $\langle \text{Adam, Carol} \rangle$ and $\langle \text{Betsy, Carol} \rangle$:

$$L = \{ \langle \text{Adam, Carol} \rangle, \langle \text{Betsy, Carol} \rangle \}$$

Above, I appealed to the idea of an ‘ordered pair’. You are probably already familiar with this concept from geometry. In geometry, you can specify a point in the 2-dimensional plane by giving its x -coordinate and its y -coordinate. So, for instance, $\langle 1, 2 \rangle$ represents the point which you reach if you go 1 unit over on the x -axis and then 2 units up

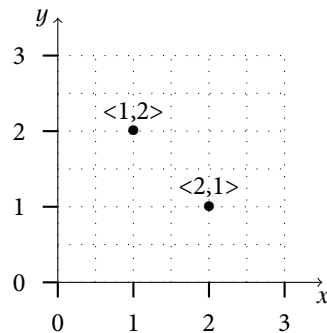


Figure 4.1

on the y -axis. The important thing to note here is that the point $\langle 1, 2 \rangle$ is very different from the point $\langle 2, 1 \rangle$. The point you reach by going 2 units over on the x -axis and then 1 unit up on the y -axis is not the same point that you reach by going 1 unit over on the x -axis and then 2 units up on the y -axis. (See figure 4.1.)

The important points to recognize and remember are just these:

1. With sets, order does not matter. $\{a, b, c\}$ is the same set as $\{b, c, a\}$.
2. With ordered pairs, order does matter. $\langle a, b \rangle$ is a different ordered pair than $\langle b, a \rangle$.

Ordered pairs are useful when we are talking about relations because, when we are talking about relations like L , order matters. Simply because Adam loves Betsy, this does not mean that Betsy loves Adam. So, just because $\langle \text{Adam}, \text{Betsy} \rangle$ is in the relation L , this should not mean that $\langle \text{Betsy}, \text{Adam} \rangle$ is in the relation L as well.

The upshot is that, rather than specifying a partial PL interpretation \mathcal{I}^P like this:

$$\mathcal{I}^P = \left\{ \begin{array}{l} \mathcal{D} = \{ \text{Adam}, \text{Betsy}, \text{Carol} \} \\ a = \text{Adam} \\ Lxy = \text{x loves y} \\ Hx = \text{x is happy} \end{array} \right.$$

We could instead provide a partial PL interpretation like so:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{ \text{Adam, Betsy, Carol} \} \\ a & = \text{Adam} \\ L & = \{ \langle \text{Adam, Carol} \rangle, \langle \text{Betsy, Carol} \rangle \} \\ H & = \{ \text{Adam} \} \end{cases}$$

4.1.4 Truth on an Interpretation

Suppose that we've got an interpretation \mathcal{I} . Then, we can lay down the following rules which tell us what the wffs of PL mean on that interpretation—that is, under which conditions they are *true* on that interpretation. (Rules (\sim) , (\vee) , $(\&)$, (\supset) , and (\equiv) should be familiar from SL .)

\mathcal{F}) A wff of the form $\lceil \mathcal{F}^n \mathbf{t}_1 \dots \mathbf{t}_n \rceil$ is true on the interpretation \mathcal{I} if the things in the domain denoted by $\lceil \mathbf{t}_1 \rceil \dots \lceil \mathbf{t}_n \rceil$ on the interpretation have the property/bear to each other the relation represented by $\lceil \mathcal{F}^n \rceil$. Otherwise, $\lceil \mathcal{F}^n \mathbf{t}_1 \dots \mathbf{t}_n \rceil$ is false on the interpretation \mathcal{I} .

Note: here, 'n' could be any number greater than or equal to one. If $n = 1$, then the way that I've written the terms, $\lceil \mathbf{t}_1 \dots \mathbf{t}_n \rceil$, is misleading, since there is only one term if $n = 1$.

- \sim) A wff of the form $\lceil \sim \mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if $\lceil \mathbf{P} \rceil$ is false on the interpretation \mathcal{I} . Otherwise, $\lceil \sim \mathbf{P} \rceil$ is false on the interpretation \mathcal{I} .
- \vee) A wff of the form $\lceil \mathbf{P} \vee \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} if either $\lceil \mathbf{P} \rceil$ is true on the interpretation \mathcal{I} or $\lceil \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} . Otherwise, $\lceil \mathbf{P} \vee \mathbf{Q} \rceil$ is false on the interpretation \mathcal{I} .
- $\&$) A wff of the form $\lceil \mathbf{P} \& \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} if both $\lceil \mathbf{P} \rceil$ is true on the interpretation \mathcal{I} and $\lceil \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} . Otherwise, $\lceil \mathbf{P} \& \mathbf{Q} \rceil$ is false on the interpretation \mathcal{I} .
- \supset) A wff of the form $\lceil \mathbf{P} \supset \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} if either $\lceil \mathbf{P} \rceil$ is false on the interpretation \mathcal{I} or $\lceil \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} . Otherwise, $\lceil \mathbf{P} \supset \mathbf{Q} \rceil$ is false on the interpretation \mathcal{I} .
- \equiv) A wff of the form $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} if both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ have the same truth value on the interpretation \mathcal{I} . Otherwise, $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is false on the interpretation \mathcal{I} .

Before getting to the rules for the quantifiers, $(\forall \mathbf{x})$ and $(\exists \mathbf{x})$, we have to introduce one more idea: that of a *variant* interpretation. We've seen that an interpretation tells us which things in the domain every variable refers to. So an interpretation will tell us that the variable x refers to some thing α in the domain, \mathcal{D} . Suppose, however, that we wish to keep everything else about the interpretation fixed, but we wish to have the variable x refer to something else, β . We may accomplish this with a variant interpretation, $\mathcal{I}_{x \rightarrow \beta}$.

If \mathcal{I} is a PL -interpretation with domain \mathcal{D} , \mathbf{x} is a variable, and α is a thing in the domain \mathcal{D} , then $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ is a PL -interpretation exactly like \mathcal{I} except that, in $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$, the variable \mathbf{x} refers to α . It is called an \mathbf{x} -variant of the interpretation \mathcal{I} .

For instance, here's a (partial) PL -interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{1, 2, 3\} \\ x & = 1 \\ y & = 2 \end{cases}$$

And here are the y -variants, $\mathcal{I}_{y \rightarrow 1}^P$, $\mathcal{I}_{y \rightarrow 2}^P$, and $\mathcal{I}_{y \rightarrow 3}^P$, of that interpretation:

$$\mathcal{I}_{y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2, 3\} \\ x & = 1 \\ y & = 1 \end{cases} \quad \mathcal{I}_{y \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2, 3\} \\ x & = 1 \\ y & = 2 \end{cases} \quad \mathcal{I}_{y \rightarrow 3}^P = \begin{cases} \mathcal{D} & = \{1, 2, 3\} \\ x & = 1 \\ y & = 3 \end{cases}$$

Notice that $\mathcal{I}_{y \rightarrow 2}^P$ is just identical to the original (partial) interpretation \mathcal{I}^P . That doesn't matter; it is still a y -variant of \mathcal{I}^P .

We're now in a position to give the rules for quantified statements being *true on an interpretation*, \mathcal{I} :

- \forall) A wff of the form $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if, for *every* α in \mathcal{D} , $\lceil \mathbf{P} \rceil$ is true on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$. Otherwise, $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ is false on the interpretation \mathcal{I} .
- \exists) A wff of the form $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if for *some* α in \mathcal{D} , $\lceil \mathbf{P} \rceil$ is true on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$. Otherwise, $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is false on the interpretation \mathcal{I} .

That is, a universally quantified wff $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ is true on an interpretation \mathcal{I} iff its immediate subformula $\lceil \mathbf{P} \rceil$ is true on the interpretation \mathcal{I} no matter what we take the

variable \mathbf{x} to refer to. And an existentially quantified wff $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is true on an interpretation \mathcal{I} iff there is at least one thing we could take \mathbf{x} to refer to which would make its immediate subformula $\lceil \mathbf{P} \rceil$ true on the interpretation \mathcal{I} .

For instance, consider the following partial interpretation, \mathcal{I}^P :

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 1 \\ y & = 2 \\ P & = \{2\} \end{cases}$$

On \mathcal{I}^P , the variable x refers to the number 1, and the variable y refers to the number 2. And ' P ' is a property which is had by the number 2. Perhaps ' P ' is the property of being prime. The interpretation doesn't give us enough information to say, since its domain is limited to the numbers 1 and 2. However, this won't end up mattering for the language PL , since the only thing that's relevant to the truth-value of a wff like ' Pa ' is whether a has the property P , and not precisely which property P is.

On this interpretation, the wff ' Px ' is false, since the thing denoted by ' x '—the number 1—does not have the property denoted by ' P '. The wff ' Py ', on the other hand, is true, since the thing denoted by the variable y —the number 2—does have the property represented by ' P '.

Even though ' Px ' and ' Py ' have different truth-values, ' $(\exists x)Px$ ' and ' $(\exists y)Py$ ' both have the same truth-value: true. To see that, note:

1. ' $(\exists x)Px$ ' is true on the interpretation \mathcal{I}^P iff there is an x -variant of \mathcal{I}^P on which ' Px ' is true.
2. $\mathcal{I}_{x \rightarrow 2}^P$ is an x -variant of \mathcal{I}^P on which x refers to the number 2. Since the number two has the property of being P , ' Px ' is true on the x -variant interpretation $\mathcal{I}_{x \rightarrow 2}^P$.
3. So, there is an x -variant of the interpretation \mathcal{I}^P on which ' Px ' is true.
4. So, ' $(\exists x)Px$ ' is true on the interpretation \mathcal{I}^P .

Similarly,

1. ' $(\exists y)Py$ ' is true on the interpretation \mathcal{I}^P iff there is a y -variant of \mathcal{I}^P on which ' Py ' is true.
2. $\mathcal{I}_{y \rightarrow 2}^P$ is a y -variant of \mathcal{I}^P on which y refers to the number 2. Since the number two has the property of being P , ' Py ' is true on the y -variant interpretation $\mathcal{I}_{y \rightarrow 2}^P$.

3. So, there is a y -variant of the interpretation \mathcal{I}^P on which ' Py ' is true.
4. So, ' $(\exists y)Py$ ' is true on the interpretation \mathcal{I}^P .

Moreover, both ' $(\forall x)Px$ ' and ' $(\forall y)Py$ ' have the same truth-value—false—on the interpretation \mathcal{I}^P .

1. ' $(\forall x)Px$ ' is true on the interpretation \mathcal{I}^P iff every x -variant of \mathcal{I}^P is one on which ' Px ' is true.
2. $\mathcal{I}_{x \rightarrow 1}^P$ is an x -variant of \mathcal{I}^P on which x refers to the number 1. Since the number 1 doesn't have the property of being P , ' Px ' is false on the x -variant interpretation $\mathcal{I}_{x \rightarrow 1}^P$.
3. So, there is an x -variant of the interpretation \mathcal{I}^P on which ' Px ' is false.
4. So, not every x -variant of \mathcal{I}^P is one which makes ' Px ' true.
5. So, ' $(\forall x)Px$ ' is false on the interpretation \mathcal{I}^P .

Similarly,

1. ' $(\forall y)Py$ ' is true on the interpretation \mathcal{I}^P iff every y -variant of \mathcal{I}^P is one on which ' Py ' is true.
2. $\mathcal{I}_{y \rightarrow 1}^P$ is a y -variant of \mathcal{I}^P on which y refers to the number 1. Since the number 1 doesn't have the property of being P , ' Py ' is false on the y -variant interpretation $\mathcal{I}_{y \rightarrow 1}^P$.
3. So, there is a y -variant of the interpretation \mathcal{I}^P on which ' Py ' is false.
4. So, not every y -variant of \mathcal{I}^P is one which makes ' Py ' true.
5. So, ' $(\forall y)Py$ ' is false on the interpretation \mathcal{I}^P .

4.1.5 Notation

A bit of new notation. Let's use expressions like

$$\mathbf{P}[\mathbf{x}], \mathbf{Q}[\mathbf{x}]$$

as variables ranging over the wffs of PL in which the variable \mathbf{x} occurs *freely* (\mathbf{x} is itself a variable ranging over the variables of PL ; it is not itself a part of the language PL). And we'll use expressions like

$$\mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}], \mathbf{Q}[\mathbf{x} \rightarrow \mathbf{t}]$$

to refer to the wffs of *PL* that you get when you replace every *free* occurrence of \mathbf{x} in $\mathbf{P}[\mathbf{x}]$ and $\mathbf{Q}[\mathbf{x}]$ with the term \mathbf{t} . That is: given a wff $\mathbf{P}[\mathbf{x}]$, you get the wff $\mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}]$ by going through $\mathbf{P}[\mathbf{x}]$, and every time \mathbf{x} appears free, you swap it out for the term \mathbf{t} .

The word ‘free’ above is important. For instance, the following is *not* a wff of *PL* in which the variable y appears freely:

$$(\forall y)(Py \ \& \ Qy)$$

While the variable ‘ y ’ appears in this wff, it does not appear freely. Both times the variable ‘ y ’ appears in ‘ $(\forall y)(Py \ \& \ Qy)$ ’, it is bound by the universal quantifier ‘ $(\forall y)$ ’. Even if a variable appears freely in a wff, that does not mean that *every* occurrence of that variable is free. So, for instance, the variable ‘ x ’ appears twice in the wff

$$Fx \supset (\exists x)Fx$$

However, while the first occurrence of ‘ x ’ is free, the second is not. The second occurrence of ‘ x ’ is bound by the existential quantifier ‘ $(\exists x)$ ’. Suppose that we use ‘ $\mathbf{P}[x]$ ’ to stand for the wff ‘ $Fx \supset (\exists x)Fx$ ’.

$$\mathbf{P}[x] = Fx \supset (\exists x)Fx$$

Then, the expression ‘ $\mathbf{P}[x \rightarrow a]$ ’ would refer to the wff ‘ $Fa \supset (\exists x)Fx$ ’, and *not* to the wff ‘ $Fa \supset (\exists x)Fa$ ’.

$$\mathbf{P}[x \rightarrow a] = Fa \supset (\exists x)Fx$$

$$\mathbf{P}[x \rightarrow a] \neq Fa \supset (\exists x)Fa$$

4.2 PL-Validity

Let’s go back to the beginning. One of the logical notions we are attempting to theorize about is the notion of an argument being *deductively valid*. We said that an *argument* is just a collection of statements, one of which is designated as the conclusion, the remainder of which are designated as the premises. And we said that an argument was *deductively valid* if and only if there is no *possibility* in which the premises are all true, yet the conclusion is false.

An argument is DEDUCTIVELY VALID if and only if there is no possibility in which the premises are all true while the conclusion is false.

In *SL*, we defined a corresponding notion of ‘*SL*-validity’ by swapping out the notion of an *argument* with the notion of an *SL*-argument, and swapping out the notion of a

possibility with the notion of a TRUTH-VALUE ASSIGNMENT. A truth-value assignment, recall, was just an assignment of truth-value (either true or false), to all of the *statement letters* of *SL*. We defined an *SL*-argument to just be a collection of wffs of *SL*, one of which is designated the conclusion, the remainder of which are labeled the premises.

A *SL*-ARGUMENT is a collection of wffs of *SL*, one of which is designated the conclusion, and the others of which are designated the premises.

And we said that an *SL*-argument is *SL*-valid if and only if there is no truth-value assignment on which the premises are all true, yet the conclusion is false.

A *SL*-argument is *SL*-VALID if and only if there is no truth-value assignment in which all of its premises are true and its conclusion is false.

In *PL*, we will do the same thing, except that we will swap of the notion of an argument with the notion of a *PL*-argument, and we will swap out the notion of a possibility with the notion of a *PL*-interpretation. A *PL*-argument is just a collection of wffs of *PL*, one of which is designated the conclusion, and the remainder of which are designated the premises.

A *PL*-ARGUMENT is a collection of wffs of *PL*, one of which is designated the conclusion, and the others of which are designated the premises.

And a *PL*-argument is *PL*-valid if and only if there is no *PL*-interpretation which makes all of its premises true while making its conclusion false.

A *PL*-argument is *PL*-VALID if and only if there is no *PL*-interpretation which makes all of its premises are true and its conclusion is false.

For instance, let's show that the following inference, called *universal instantiation (UI)*, is *PL*-valid:

$$\frac{(\forall \mathbf{x})\mathbf{P}[\mathbf{x}]}{\mathbf{P}[\mathbf{x} \rightarrow \mathbf{a}]}$$

Pick any interpretation, \mathcal{I} , which makes the premise, $\lceil (\forall \mathbf{x})\mathbf{P}[\mathbf{x}] \rceil$, true. In the interpretation \mathcal{I} , the constant $\lceil \mathbf{a} \rceil$ must refer to something in interpretation's domain, \mathcal{D} . Call that thing—whatever it is— $\lceil \alpha \rceil$. Now, a wff of the form $\lceil (\forall \mathbf{x})\mathbf{P}[\mathbf{x}] \rceil$ is true on an

interpretation \mathcal{I} if and only if $\lceil \mathbf{P}[\mathbf{x}] \rceil$ is true on every \mathbf{x} -variant interpretation. $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ is an \mathbf{x} -variant of \mathcal{I} . So, if $\lceil (\forall \mathbf{x})\mathbf{P}[\mathbf{x}] \rceil$ is true on \mathcal{I} , then $\lceil \mathbf{P}[\mathbf{x}] \rceil$ must be true on the variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$. But, on the variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$, $\lceil \mathbf{x} \rceil$ refers to the very same thing as $\lceil \mathbf{a} \rceil$ refers to on the interpretation \mathcal{I} —that is, α . Otherwise, $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ is exactly the same as \mathcal{I} . So, if $\lceil \mathbf{P}[\mathbf{x}] \rceil$ is true on the interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$, then $\lceil \mathbf{P}[\mathbf{x} \rightarrow \mathbf{a}] \rceil$ must be true on the interpretation \mathcal{I} .

Therefore, every *PL*-interpretation which makes a wff of the form $\lceil (\forall \mathbf{x})\mathbf{P}[\mathbf{x}] \rceil$ true makes a wff of the form $\lceil \mathbf{P}[\mathbf{x} \rightarrow \mathbf{a}] \rceil$ true as well. So *UI* is *PL*-valid. The kind of informal proof I just provided is known as a *semantic proof*. I will have more to say about these kinds of informal proofs below.

An argument of *PL* is *PL-INVALID* if and only if it is *not PL*-valid. That is:

A *PL*-argument is *PL-INVALID* if and only if there is some *PL*-interpretation on which its premises are all true and its conclusions is false.

We may introduce the notion of a *PL-COUNTEREXAMPLE*. This is just a *PL*-interpretation which makes all of the premises of a *PL*-argument true, yet makes its conclusion false.

A *PL-COUNTEREXAMPLE* to a *PL*-argument is a *PL*-interpretation on which the argument's premises are all true and its conclusion is false.

With this notion of a *PL*-counterexample in hand, we may give a simpler definition of *PL*-validity and *PL*-invalidity. An argument of *PL* is *PL*-valid if and only if it has no *PL*-counterexample. And it is *PL*-invalid if and only if it has a *PL*-counterexample.

An argument of *PL* is *PL-VALID* if and only if it has no *PL*-counterexample.

An argument of *PL* is *PL-INVALID* if and only if it has some *PL*-counterexample.

Therefore, to show that an argument is *PL-INVALID*, you can just provide a *PL*-counterexample. That is: you may just provide a *PL*-interpretation on which the premises of the argument are true, yet the conclusion is false.

For instance, to show that the following argument is *PL*-invalid:

$$\frac{(\exists x)Ax \ \& \ (\exists x)Bx}{(\exists x)(Ax \ \& \ Bx)}$$

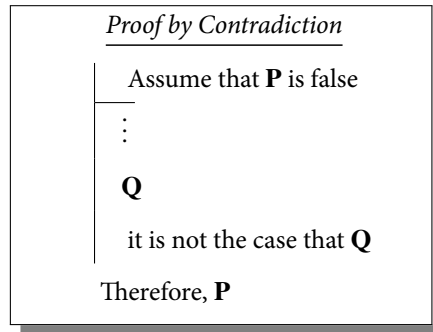
It suffices to provide the following (partial) *PL*-interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ A & = \{1\} \\ B & = \{2\} \end{cases}$$

Given this partial *PL*-interpretation, the premise of the above argument is true, yet its conclusion is false. So, it provides a *PL*-counterexample to the validity of the argument $(\exists x)Ax \ \& \ (\exists x)Bx \ // \ (\exists x)(Ax \ \& \ Bx)$.

4.3 Proving *PL*-validity and the method of Semantic Proof

Suppose that we wish to show that an argument of *PL* is *PL*-valid. Here is a way to do that: we utilize the proof method known as *reductio ad absurdum*, or “proof by contradiction”. In a proof by contradiction, you assume the negation of the thing that you wish to show, and derive a contradiction from it. Since you’ve shown that your assumption leads to a contradiction, you can conclude that your assumption must be false.



In the above, I’ve drawn scope lines to indicate that all of the reasoning taking place where the ‘⋮’ occur is taking place under the assumption that **P** is false. However, we need not include explicit scope lines when we are providing an informal proof. Nevertheless, we should keep in mind during such a proof which steps are taking place within the scope of our assumption, and which are not.

We can utilize this proof technique to show that an argument is valid by supposing that it is invalid and then deriving a contradiction. In deriving the contradiction, we will have to make use of what we know about the semantics of the wffs of *PL*.

For instance, if we wish to show that $\sim(\forall x)Fx // (\exists x)\sim Fx$ is *PL*-valid, we could begin by assuming that there is some interpretation which makes the premise true and the

conclusion false, and then deriving a contradiction. We would proceed as follows:

- 1) Assume that there is some interpretation, \mathcal{I} , which makes ' $\sim(\forall x)Fx$ ' true, but which makes ' $(\exists x)\sim Fx$ ' false.
- 2) Thus, \mathcal{I} makes ' $(\forall x)Fx$ ' false. [from (1) and the definition of ' \sim ']
- 3) ' $(\forall x)Fx$ ' is false on the interpretation \mathcal{I} if and only if there is some x -variant of \mathcal{I} which makes ' Fx ' true. [by the definition of ' \forall ']
- 4) Therefore, there is some x -variant of \mathcal{I} — call it ' $\mathcal{I}_{x \rightarrow \alpha}$ ' — which makes ' Fx ' false. [from (2) and (3)]
- 5) $\mathcal{I}_{x \rightarrow \alpha}$ makes ' Fx ' false iff it makes ' $\sim Fx$ ' true. [by the definition of ' \sim ']
- 6) So, $\mathcal{I}_{x \rightarrow \alpha}$ makes ' $\sim Fx$ ' true. [by (4) and (5)]
- 7) So, there is an x -variant of the interpretation \mathcal{I} which makes ' $\sim Fx$ ' true. [from line (6)]
- 8) ' $(\exists x)\sim Fx$ ' false on the interpretation \mathcal{I} [from line (1)]
- 9) ' $(\exists x)\sim Fx$ ' false on the interpretation \mathcal{I} if and only if there is no x -variant of the interpretation \mathcal{I} which makes ' $\sim Fx$ ' true. [from the definition of ' \exists ']
- 10) So, there is no x -variant of the interpretation \mathcal{I} which makes ' $\sim Fx$ ' true. [from (8) and (9)].
- 11) But there is an x -variant of the interpretation \mathcal{I} which makes ' $\sim Fx$ ' true. [from (7)].
- 12) Because our assumption that there is some interpretation, \mathcal{I} , which makes ' $\sim(\forall x)Fx$ ' true, but which makes ' $(\exists x)\sim Fx$ ' false led to a contradiction (lines (10) and (11)), that assumption must be false.
- 13) Therefore, there is no interpretation which makes ' $\sim(\forall x)Fx$ ' true, but which makes ' $(\exists x)\sim Fx$ ' false. [from (12)]
- 14) Thus, $\sim(\forall x)Fx // (\exists x)\sim Fx$ is *PL*-valid [from (13) and the definition of *PL*-validity]

The proof in the boxed text above is a *semantic proof*. It is similar to the kinds of derivations (or 'deductions') you learned about in PHIL 0500. It begins with an assumption, and every step from there on out proceeds by citing the previous lines and certain facts

about the semantics for PL . However, this is not a derivation within any formal system. With these semantic proofs, we use reasoning techniques which we may be familiar with from a formal derivation system to reason *about* the language PL itself.

4.4 Translations from PL into English

In order to translate from PL into English, we will first need a (partial) PL -interpretation. This interpretation will tell us the meanings of the predicates of PL as well as the constants and (free) variables of PL . To refresh your memory, a partial PL -interpretation will provide the following things:

A partial PL -INTERPRETATION, \mathcal{I}^P , of a wff or set of wffs of PL provides:

1. A specification of which things fall in the *domain*, \mathcal{D} , of the interpretation.
2. A specification of which things in the domain the *constants* appearing in the wff or wffs of PL represent.
3. A specification of which things in the domain the *free variables* appearing in the wff or wffs of PL represent.
4. For every n -place predicate appearing in the wff or wffs of PL , a specification of the n -place property it represents.

4.4.1 Translating Atomic wffs of PL

Call a wff of PL 'atomic' iff it consists of only an n -place predicate followed by n terms. Our partial interpretation will tell us which property or relation a given predicate of PL represents, and it will tell us which objects in the domain the terms of PL refer to. We translate atomic wffs of PL into English by just saying that the things denoted by the terms have the property or bear the relation to one another denoted by the predicate.

For instance, suppose that we have the following partial *PL*-interpretation:

$$\mathcal{I}^p = \left\{ \begin{array}{l} \mathcal{D} = \text{the set of all concrete things in the Garden of Eden} \\ a = \text{Adam} \\ e = \text{Eve} \\ p = \text{the apple} \\ L\mathbf{x}\mathbf{y} = \mathbf{x} \text{ loves } \mathbf{y} \\ P\mathbf{x} = \mathbf{x} \text{ has parents} \\ G\mathbf{x}\mathbf{y}\mathbf{z} = \mathbf{x} \text{ gave } \mathbf{y} \text{ to } \mathbf{z} \end{array} \right.$$

(Note: it's important that the stuff referred to in the domain really exist—if you don't think that the Garden of Eden exists, then take us to be talking about the fiction in which it does.) Then, the we may translate the following wffs of *PL* into English as follows:

$$\begin{aligned} Pa &= \text{Adam has parents.} \\ Pe &= \text{Eve has parents.} \\ Lae &= \text{Adam loves Eve} \\ Laa &= \text{Adam loves himself} \\ Gepa &= \text{Eve gave the apple to Adam.} \end{aligned}$$

Once we've translated the atomic wffs of *PL*, we know how to translate sentences involving the atomic wffs and \sim , $\&$, \vee , \supset , and \equiv . That's what we learned how to do in *SL*. So, on the partial *PL*-interpretation above, we can translate the following wffs of *PL* as follows:

$$\begin{aligned} \sim Pa &= \text{Adam doesn't have parents.} \\ \sim(Pa \vee Pe) &= \text{Neither Adam nor Eve has parents.} \\ Lae \equiv Lea &= \text{Adam loves Eve if and only if Eve loves Adam.} \\ Laa \supset \sim Gapa &= \text{If Adam loves himself, then he doesn't give the apple to himself} \\ Gepa \supset \sim Gape &= \text{If Eve gave the apple to Adam, then Adam didn't give the apple to Eve.} \end{aligned}$$

4.4.2 Translating Simple Quantified wffs of *PL*

We already know how to translate expressions involving the operators \sim , $\&$, \vee , \supset , and \equiv into English. What's needed is a method for translating the quantifiers $(\forall \mathbf{x})$ and $(\exists \mathbf{x})$

into English. The following will do as a good translation guide in the simple case where the quantifier scopes over a wff consisting of just an 1-place predicate followed by 1 bound variable.

$$\begin{aligned} (\forall \mathbf{x})\mathcal{F}\mathbf{x} &\longrightarrow \text{Everything (in the domain) is } \mathcal{F}. \\ (\exists \mathbf{x})\mathcal{F}\mathbf{x} &\longrightarrow \text{Something (in the domain) is } \mathcal{F}. \end{aligned}$$

For instance, given the partial interpretation \mathcal{I}^P ,

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \text{the set of all works of art} \\ B\mathbf{x} & = \mathbf{x} \text{ is beautiful} \end{cases}$$

We can give the following translations from *PL* into English:

$$\begin{aligned} (\forall y)By &\longrightarrow \text{Every work of art is beautiful.} \\ (\exists z)Bz &\longrightarrow \text{Some work of art is beautiful.} \end{aligned}$$

4.4.3 Translating More Complicated Quantified wffs of *PL*

Often, a quantified wff of *PL* will have a more complicated wff in its scope. There are four kinds of quantified wffs of *PL* that you should be familiar with, and which you should be able to translate from *PL* to English (and *vice versa*). These are known as the **A**, **E**, **I**, and **O** sentence forms.

$$\begin{aligned} (\forall \mathbf{x})(\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x}) &\longrightarrow \text{All } \mathcal{S} \text{ are } \mathcal{P} && \text{(A)} \\ (\forall \mathbf{x})(\mathcal{S}\mathbf{x} \supset \sim \mathcal{P}\mathbf{x}) &\longrightarrow \text{No } \mathcal{S} \text{ are } \mathcal{P}. && \text{(E)} \\ (\exists \mathbf{x})(\mathcal{S}\mathbf{x} \& \mathcal{P}\mathbf{x}) &\longrightarrow \text{Some } \mathcal{S} \text{ are } \mathcal{P}. && \text{(I)} \\ (\exists \mathbf{x})(\mathcal{S}\mathbf{x} \& \sim \mathcal{P}\mathbf{x}) &\longrightarrow \text{Some } \mathcal{S} \text{ are not } \mathcal{P}. && \text{(O)} \end{aligned}$$

Some \mathcal{S} are \mathcal{P}

To see why these wffs of *PL* translate into these English sentences, we should think about how to represent the content of the English sentences in terms of Venn diagrams. The way to represent a sentence of the form ‘Some \mathcal{S} are \mathcal{P} ’ with a Venn Diagram is as shown in figure 4.2. That is, ‘Some \mathcal{S} are \mathcal{P} ’ is true if and only if there is something which is both \mathcal{S} and \mathcal{P} —*i.e.*, if and only if there is something which is inside both of the circles \mathcal{S} and \mathcal{P} . There will be something like that—call it ‘ α ’—if and only if there is some \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ on which the wff ‘ $\mathcal{S}\mathbf{x} \& \mathcal{P}\mathbf{x}$ ’ is true. But there will be a \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ on which the wff ‘ $\mathcal{S}\mathbf{x} \& \mathcal{P}\mathbf{x}$ ’ is true if and only if ‘ $(\exists \mathbf{x})(\mathcal{S}\mathbf{x} \& \mathcal{P}\mathbf{x})$ ’ is true, since (from above):

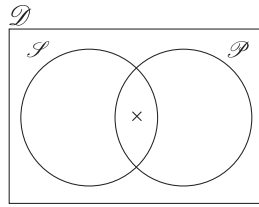


Figure 4.2:

\exists) A wff of the form $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if there is *some* \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$, where α is in the domain \mathcal{D} of \mathcal{I} , which makes \mathbf{P} true. Otherwise, it is false on the interpretation \mathcal{I} .

So $\lceil \text{Some } \mathcal{S} \text{ are } \mathcal{P} \rceil$ is a good translation of $\lceil (\exists \mathbf{x})(\mathcal{S}\mathbf{x} \ \& \ \mathcal{P}\mathbf{x}) \rceil$ (and *vice versa*).

Some \mathcal{S} are not \mathcal{P}

Next, consider $\lceil \text{Some } \mathcal{S} \text{ are not } \mathcal{P} \rceil$. The way to represent a sentence like this with a Venn diagram is as shown in figure 4.3. That is, $\lceil \text{Some } \mathcal{S} \text{ are not } \mathcal{P} \rceil$ is true if and

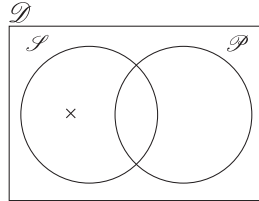


Figure 4.3:

only if there is something which is \mathcal{S} but not \mathcal{P} —*i.e.*, if and only if there is something which is inside the circle \mathcal{S} yet outside of the circle \mathcal{P} . There will be something like that—call it ' α '—if and only if there is some \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ on which the wff $\lceil \mathcal{S}\mathbf{x} \ \& \ \sim \mathcal{P}\mathbf{x} \rceil$ is true. But there will be a \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ on which the wff $\lceil \mathcal{S}\mathbf{x} \ \& \ \sim \mathcal{P}\mathbf{x} \rceil$ is true if and only if $\lceil (\exists \mathbf{x})(\mathcal{S}\mathbf{x} \ \& \ \sim \mathcal{P}\mathbf{x}) \rceil$ is true, since (again):

\exists) A wff of the form $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if there is *some* \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$, where α is in the domain \mathcal{D} of \mathcal{I} , which makes \mathbf{P} true. Otherwise, it is false on the interpretation \mathcal{I} .

So $\lceil \text{Some } \mathcal{S} \text{ are not } \mathcal{P} \rceil$ is a good translation of $\lceil (\exists \mathbf{x})(\mathcal{S}\mathbf{x} \ \& \ \sim \mathcal{P}\mathbf{x}) \rceil$ (and *vice versa*).

All \mathcal{S} are \mathcal{P}

Next, consider 'All \mathcal{S} are \mathcal{P} '. The way to represent a sentence like this with a Venn diagram is as shown in figure 4.4. That is, 'All \mathcal{S} are \mathcal{P} ' is true if and only if there

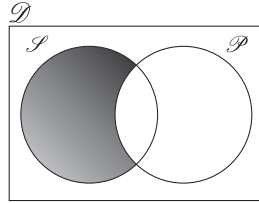


Figure 4.4:

is nothing which is \mathcal{S} but not \mathcal{P} —i.e., if and only if there is nothing which is inside the circle \mathcal{S} yet outside of the circle \mathcal{P} . Think about what it would take for this claim to be *false*. This claim would be false if and only if there were something which were \mathcal{S} but not \mathcal{P} . Otherwise, it would be true. Suppose that there were something—call it ' α '—which were \mathcal{S} but not \mathcal{P} . Then, the wff of PL $\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x}$ would be false on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ —since its antecedent is true, yet its consequent is false. On the other hand, if anything β in the domain is either both \mathcal{S} and \mathcal{P} or not \mathcal{S} , then $\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x}$ would still be true on the variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \beta}$ (by the definition of ' \supset ').

So, there is something which is \mathcal{S} and not \mathcal{P} if and only if there is some α in the domain such that $\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x}$ is false when \mathbf{x} refers to α .

So, there is something which is \mathcal{S} and not \mathcal{P} if and only if ' $(\forall \mathbf{x})(\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x})$ ' is false, since (from before):

- \forall) A wff of the form ' $(\forall \mathbf{x})\mathbf{P}$ ' is true on the interpretation \mathcal{I} if, for every α in the domain of \mathcal{I} , ' \mathbf{P} ' is true on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$. Otherwise, it is false on the interpretation \mathcal{I} .

By the same token, if there is nothing which is \mathcal{S} and not \mathcal{P} , then ' $(\forall \mathbf{x})(\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x})$ ' will be true, since ' $\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x}$ ' will be true on all \mathbf{x} -variant interpretations.

So 'All \mathcal{S} are \mathcal{P} ' is true in exactly the same circumstances as ' $(\forall \mathbf{x})(\mathcal{S}\mathbf{x} \supset \mathcal{P}\mathbf{x})$ '. So the former provides a good translation of the latter (and *vice versa*).

No \mathcal{S} are \mathcal{P}

Finally, consider ‘No \mathcal{S} are \mathcal{P} ’. We saw that the way to represent a sentence like this with a Venn diagram is as shown in figure 4.5. That is: the claim ‘No \mathcal{S} are \mathcal{P} ’ is

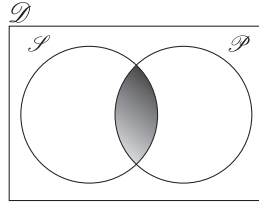


Figure 4.5:

true if and only if there is nothing which is both \mathcal{S} and \mathcal{P} . Think about the circumstances under which this claim would be *false*. It would be false if and only if there were something—call it ‘ α ’—which were both \mathcal{S} and \mathcal{P} . Then, on the variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ ‘ $\mathcal{S}\mathbf{x}$ ’ would be true and ‘ $\sim\mathcal{P}\mathbf{x}$ ’ would be false. So ‘ $\mathcal{S}\mathbf{x} \supset \sim\mathcal{P}\mathbf{x}$ ’ would be false (by the definition of ‘ \supset ’). So ‘ $(\forall\mathbf{x})(\mathcal{S}\mathbf{x} \supset \sim\mathcal{P}\mathbf{x})$ ’ would be false (since it is false on the variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$).

If there were nothing which were both \mathcal{S} and \mathcal{P} , then ‘No \mathcal{S} are \mathcal{P} ’ would be true. And, similarly, ‘ $(\forall\mathbf{x})(\mathcal{S}\mathbf{x} \supset \sim\mathcal{P}\mathbf{x})$ ’ would be true, since the only way that could be false would be if there were an \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ which made

$$\mathcal{S}\mathbf{x} \supset \sim\mathcal{P}\mathbf{x}$$

false. But the above wff would be false on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$ only if α were both \mathcal{S} and \mathcal{P} —since that is the only thing that would make its antecedent true and its consequent false.

So ‘No \mathcal{S} are \mathcal{P} ’ is true in exactly the same circumstances as ‘ $(\forall\mathbf{x})(\mathcal{S}\mathbf{x} \supset \sim\mathcal{P}\mathbf{x})$ ’. So the former provides a good translation of the latter (and *vice versa*).

4.5 Translations from English into *PL*

The English expressions appearing in the translation guides from the previous section constitute the *canonical logical form* of English. In general, if we have an English expression in canonical logical form, we may translate it into *PL* directly according to that

translation schema:

Everything is \mathcal{F}	\rightarrow	$(\forall \mathbf{x}) \mathcal{F} \mathbf{x}$
Something is \mathcal{F}	\rightarrow	$(\exists \mathbf{x}) \mathcal{F} \mathbf{x}$
Some \mathcal{S} are \mathcal{P}	\rightarrow	$(\exists \mathbf{x})(\mathcal{S} \mathbf{x} \ \& \ \mathcal{P} \mathbf{x})$
Some \mathcal{S} are not \mathcal{P}	\rightarrow	$(\exists \mathbf{x})(\mathcal{S} \mathbf{x} \ \& \ \sim \mathcal{P} \mathbf{x})$
All \mathcal{S} are \mathcal{P}	\rightarrow	$(\forall \mathbf{x})(\mathcal{S} \mathbf{x} \supset \mathcal{P} \mathbf{x})$
No \mathcal{S} are \mathcal{P}	\rightarrow	$(\forall \mathbf{x})(\mathcal{S} \mathbf{x} \supset \sim \mathcal{P} \mathbf{x})$

There are, however, other ways of translating these English sentences into *PL*. For instance, given the following (partial) interpretation,

$$\mathcal{S}^{\mathcal{P}} = \begin{cases} \mathcal{D} & = \text{the set of all people} \\ R\mathbf{x} & = \mathbf{x} \text{ is a Republican} \\ S\mathbf{x} & = \mathbf{x} \text{ is socially liberal} \end{cases}$$

each of the following wffs of *PL* correctly translate the English sentence ‘Some Republicans are socially liberal’.

$$\text{Some Republicans are socially liberal} \rightarrow \begin{cases} (\exists x)(Rx \ \& \ Sx) \\ \sim(\forall x)(Rx \supset \sim Sx) \end{cases}$$

These wffs are *PL*-equivalent—they are true in all the same *PL*-interpretations and false in all the same *PL*-interpretations.

Likewise, each of the following wffs correctly translate the English ‘Some Republicans are not socially liberal’:

$$\text{Some Republicans are not socially liberal} \rightarrow \begin{cases} (\exists x)(Rx \ \& \ \sim Sx) \\ \sim(\forall x)(Rx \supset Sx) \end{cases}$$

These wffs are *PL*-equivalent—they are true in all the same *PL*-interpretations and false in all the same *PL*-interpretations.

Also, each of the following wffs of *PL* correctly translate the English sentence ‘No Republicans are socially liberal’.

$$\text{No Republicans are socially liberal} \rightarrow \begin{cases} (\forall x)(Rx \supset \sim Sx) \\ \sim(\exists x)(Rx \ \& \ Sx) \end{cases}$$

These wffs are *PL*-equivalent—they are true in all the same *PL*-interpretations and false

in all the same *PL*-interpretations.

Similarly, each of the following wffs correctly translate the English ‘All Republicans are socially liberal’:

$$\text{All Republicans are socially liberal} \longrightarrow \begin{cases} (\forall x)(Rx \supset Sx) \\ \sim(\exists x)(Rx \& \sim Sx) \end{cases}$$

These wffs are *PL*-equivalent—they are true in all the same *PL*-interpretations and false in all the same *PL*-interpretations.

4.6 Overlapping Quantifiers and Relational Predicates

We may define *PL*-Equivalence in exactly the same way that we defined *SL*-Equivalence, except that, instead of focusing on *truth-value assignments*, as we did in *SL*,

Two wffs of *SL*, ‘**P**’ and ‘**Q**’, are *SL*-equivalent if and only if there is no truth-value assignment on which ‘**P**’ and ‘**Q**’ have different truth-values.

we will instead take the relevant notion of possibility to be given by *PL*-INTERPRETATIONS. Thus:

Two wffs of *PL*, ‘**P**’ and ‘**Q**’, are *PL*-equivalent if and only if there is no *PL*-interpretation on which ‘**P**’ and ‘**Q**’ have different truth-values.

Thus, if we wish to show that two wffs of *PL* are *not PL*-equivalent, it suffices to provide a single *PL*-interpretation on which one of the wffs is true while the other is false.

Today, let’s consider whether changing the order of quantifiers or the order of bound variables (or changing both) makes a difference to the meaning of wffs of *PL* by asking whether two wffs which differ only in the order of their quantifiers, the order of their bound variables, or both the order of their quantifiers and the order of their bound variables, are *PL*-equivalent or not.

4.6.1 Changing the Order of the Quantifiers

Consider the following partial interpretation:

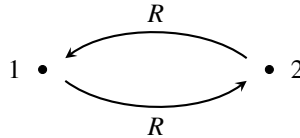
$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

If you want a more familiar way to think about the relation R , you could think about it like this:

$$Rxy = (x \neq y)$$

Thus, because $1 \neq 2$ and $2 \neq 1$, 1 bears the relation R to 2 and 2 bears the relation R to 1. However, since $1 = 1$ and $2 = 2$, 1 does not bear the relation R to itself and 2 does not bear the relation R to itself. However, the only thing we need to know about R is this: 1 bears it to 2, 2 bears it to 1, 1 doesn't bear it to 1, and 2 doesn't bear it to 2.

If it helps you, you can think about the relation R with the following picture:



In the picture, if a thing α shows up at the base of an arrow labeled ' R ' and a thing β shows up at the head of the arrow, then α bears the relation R to β . So, the above picture just tells us that 1 bears the relation R to 2 and 2 bears the relation R to 1, and nothing else bears the relation R to anything else.

Now consider the following two wffs of PL :

$$(\forall x)(\exists y)Rxy$$

$$(\exists y)(\forall x)Rxy$$

Let's think about what these wffs say on the partial interpretation \mathcal{I}^P .

$$(\forall x)(\exists y)Rxy$$

Start with ‘ $(\forall x)(\exists y)Rxy$ ’. This wff is true iff ‘ $(\exists y)Rxy$ ’ is true on every x -variant of \mathcal{I}^P . There are two x -variants of \mathcal{I}^P :

$$\mathcal{I}_{x \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 1 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{x \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

Start with $\mathcal{I}_{x \rightarrow 1}^P$. Is ‘ $(\exists y)Rxy$ ’ true on this partial interpretation? Well, ‘ $(\exists y)Rxy$ ’ is true according to $\mathcal{I}_{x \rightarrow 1}^P$ if and only if ‘ Rxy ’ is true according to some y -variant of the interpretation $\mathcal{I}_{x \rightarrow 1}^P$. There are two y -variants of this interpretation:

$$\mathcal{I}_{x \rightarrow 1, y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 1 \\ y & = 1 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{x \rightarrow 1, y \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 1 \\ y & = 2 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

On the first, $\mathcal{I}_{x \rightarrow 1, y \rightarrow 1}^P$, ‘ Rxy ’ is false, since 1 does not bear the relation R to itself. On the second, $\mathcal{I}_{x \rightarrow 1, y \rightarrow 2}^P$, ‘ Rxy ’ is true, since 1 does bear the relation R to 2. All that matters, however, is that ‘ Rxy ’ is true on at least one of these interpretations. Since it is, ‘ $(\exists y)Rxy$ ’ is true on the interpretation $\mathcal{I}_{x \rightarrow 1}^P$.

What about the second x -variant interpretation $\mathcal{I}_{x \rightarrow 2}^P$? Well, ‘ $(\exists y)Rxy$ ’ is true on this interpretation iff ‘ Rxy ’ is true on some y -variant of it. There are two y -variants of this interpretation:

$$\mathcal{I}_{x \rightarrow 2, y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ y & = 1 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{x \rightarrow 2, y \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ y & = 2 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

On the first, $\mathcal{I}_{x \rightarrow 2, y \rightarrow 1}^P$, ‘ Rxy ’ is true, since 2 bears the relation R to 1. On the second, $\mathcal{I}_{x \rightarrow 2, y \rightarrow 2}^P$, ‘ Rxy ’ is false, since 2 does not bear the relation R to itself. All that matters, however, is that ‘ Rxy ’ is true at *at least one* y -variant of the interpretation $\mathcal{I}_{x \rightarrow 2}^P$. Since it is, ‘ $(\exists y)Rxy$ ’ is true on the interpretation ‘ $\mathcal{I}_{x \rightarrow 2}^P$ ’.

Thus, ‘ $(\exists y)Rxy$ ’ is true on the interpretation $\mathcal{I}_{x \rightarrow 1}^P$ and ‘ $(\exists y)Rxy$ ’ is true on the interpretation $\mathcal{I}_{x \rightarrow 2}^P$. Thus, ‘ $(\exists y)Rxy$ ’ is true on every x -variant of \mathcal{I}^P . Thus, ‘ $(\forall x)(\exists y)Rxy$ ’ is true on the interpretation \mathcal{I}^P .

$$(\exists y)(\forall x)Rxy$$

Now let's think about the wff $(\exists y)(\forall x)Rxy$. This wff is exactly like the last wff we considered— $(\forall x)(\exists y)Rxy$ —except that we have swapped the order of the universal and the existential quantifiers. Does this make a difference to the truth-value of the wff? Let us check and see.

$(\exists y)(\forall x)Rxy$ is true on \mathcal{I}^P if and only if $(\forall x)Rxy$ is true on some y -variant of \mathcal{I}^P . There are two y -variants of \mathcal{I}^P :

$$\mathcal{I}_{y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ y & = 1 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{y \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ y & = 2 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

Let's start with the first— $\mathcal{I}_{y \rightarrow 1}^P$. Is $(\forall x)Rxy$ true on this interpretation? Well, $(\forall x)Rxy$ is true on $\mathcal{I}_{y \rightarrow 1}^P$ iff Rxy is true on every x -variant of $\mathcal{I}_{y \rightarrow 1}^P$. There are two x -variants of $\mathcal{I}_{y \rightarrow 1}^P$:

$$\mathcal{I}_{y \rightarrow 1, x \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ y & = 1 \\ x & = 1 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{y \rightarrow 1, x \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ y & = 1 \\ x & = 2 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

On $\mathcal{I}_{y \rightarrow 1, x \rightarrow 1}^P$, Rxy is false, since 1 does not bear the relation R to itself. Thus, there is some x -variant of $\mathcal{I}_{y \rightarrow 1}^P$ on which Rxy is false. So it is not the case that Rxy is true on every x -variant of $\mathcal{I}_{y \rightarrow 1}^P$. So $(\forall x)Rxy$ is false on the interpretation $\mathcal{I}_{y \rightarrow 1}^P$.

This doesn't yet show that $(\exists y)(\forall x)Rxy$ is false on the interpretation \mathcal{I}^P , since all that it takes for this wff to be true on \mathcal{I}^P is for there to be *some* y -variant of it on which $(\forall x)Rxy$ is true. So let us consider the second y -variant interpretation: $\mathcal{I}_{y \rightarrow 2}^P$.

$(\forall x)Rxy$ is true on $\mathcal{I}_{y \rightarrow 2}^P$ iff Rxy is true on every x -variant of this interpretation. There are two x -variants of this interpretation:

$$\mathcal{I}_{y \rightarrow 2, x \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ y & = 2 \\ x & = 1 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{y \rightarrow 2, x \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ y & = 2 \\ x & = 2 \\ R & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

On the first, Rxy is true, since 2 does bear the relation R to 1. However, on the second, Rxy is false, since 2 does not bear the relation R to itself. Thus, Rxy is not true on

every x -variant of $\mathcal{I}_{y \rightarrow 2}^P$. So $'(\forall x)Rxy'$ is false on the interpretation $\mathcal{I}_{y \rightarrow 2}^P$.

Therefore, $'(\forall x)Rxy'$ is false on every y -variant of \mathcal{I}^P . So $'(\exists y)(\forall x)Rxy'$ is false on the interpretation \mathcal{I}^P .

So, \mathcal{I}^P is an interpretation on which $'(\forall x)(\exists y)Rxy'$ is true, yet $'(\exists y)(\forall x)Rxy'$ is false. So $'(\forall x)(\exists y)Rxy'$ and $'(\exists y)(\forall x)Rxy'$ are not *PL*-equivalent. So the order of the quantifiers does make a difference to the meaning of a quantified wff of *PL*.

Translating $'(\forall x)(\exists y)Rxy'$ and $'(\exists y)(\forall x)Rxy'$

On the interpretation \mathcal{I}^P , the first wff, $'(\forall x)(\exists y)Rxy'$, says 'Every number in the domain has some number in the domain that it's distinct from (not identical to)'. And this is true: 1 is distinct from 2, and 2 is distinct from 1. So every number in the domain has some number in the domain that it's distinct from.

On the same interpretation, the second wff, $'(\exists y)(\forall x)Rxy'$ says 'There's some number in the domain which is distinct from every number in the domain'. However, this is false. 1 is not distinct from itself, so 1 is not distinct from every number in the domain. And 2 is not distinct from itself, so 2 is not distinct from every number in the domain. So there is no number which is distinct from every number in the domain.

Consider the following (partial) interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \text{the set of all people} \\ Axy & = \text{x admires y} \end{cases}$$

And consider the following two wffs:

$$\begin{aligned} &(\forall x)(\exists y)Axy \\ &(\exists y)(\forall x)Axy \end{aligned}$$

The first wff says that everybody has somebody that they admire. The second wff says that somebody is such that everybody admires *them*. Now, suppose that everybody admires their parents and nobody else. Then, the first wff would be true (since everybody does have somebody they admire—namely, their parents). However, the second wff would be false (since there is nobody who is everybody's parents, there is nobody who everybody admires).

The lesson: changing the order of the quantifiers can make a difference to the meaning and the truth-value of a wff of *PL*.

4.6.2 Changing the Order of the Bound Variables

What about changing the order of the bound *variables*? Does that make a difference to the truth-value of a wff like $(\forall x)(\exists y)Rxy$? It does. Consider the following pair of wffs:

$$\begin{aligned} &(\forall x)(\exists y)Rxy \\ &(\forall x)(\exists y)Ryx \end{aligned}$$

If we wish to show that these two wffs have a different meaning—that is, that they are not *PL*-equivalent—then it suffices to provide a single (partial) *PL*-interpretation on which the wffs have different truth-values. Consider the following (partial) interpretation:

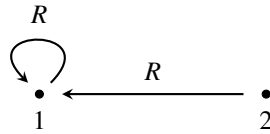
$$\mathcal{I}^P = \begin{cases} \mathcal{D} &= \{1, 2\} \\ R &= \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

If you want a more familiar way to think about the relation R , you could think about it like this:

$$Rxy = (x \times y = x)$$

Thus, because $1 \times 1 = 1$ and $2 \times 1 = 2$, 1 bears the relation R to itself and 2 bears the relation R to 1. However, since $1 \times 2 \neq 1$ and $2 \times 2 \neq 2$, 1 does not bear the relation R to 2 and 2 does not bear the relation R to itself. However, the only thing we need to know about R is this: 1 bears it to 1, 2 bears it to 1, 1 doesn't bear it to 2, and 2 doesn't bear it to 2.

If it helps you, you can think about the relation R with the following picture:



$$(\forall x)(\exists y)Rxy$$

$(\forall x)(\exists y)Rxy$ is true on the interpretation \mathcal{I}^P iff $(\exists y)Rxy$ is true on every x -variant of \mathcal{I}^P . There are two x -variants of \mathcal{I}^P :

$$\mathcal{I}_{x \rightarrow 1}^P = \begin{cases} \mathcal{D} &= \{1, 2\} \\ x &= 1 \\ R &= \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{x \rightarrow 2}^P = \begin{cases} \mathcal{D} &= \{1, 2\} \\ x &= 2 \\ R &= \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

Take $\mathcal{I}_{x \rightarrow 1}^P$ first. ' $(\exists y)Rxy$ ' is true on $\mathcal{I}_{x \rightarrow 1}^P$ iff ' Rxy ' is true on some y -variant of $\mathcal{I}_{x \rightarrow 1}^P$. Consider, however, the following y -variant of $\mathcal{I}_{x \rightarrow 1}^P$:

$$\mathcal{I}_{x \rightarrow 1, y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 1 \\ y & = 1 \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

On this interpretation, ' Rxy ' is true, since 1 bears the relation R to itself. Thus, ' Rxy ' is true on some y -variant of the interpretation $\mathcal{I}_{x \rightarrow 1}^P$. So ' $(\exists y)Rxy$ ' is true on $\mathcal{I}_{x \rightarrow 1}^P$.

Consider next the x -variant interpretation $\mathcal{I}_{x \rightarrow 2}^P$. ' $(\exists y)Rxy$ ' is true on $\mathcal{I}_{x \rightarrow 2}^P$ iff ' Rxy ' is true on some y -variant of $\mathcal{I}_{x \rightarrow 2}^P$. Consider, however, the following y -variant of $\mathcal{I}_{x \rightarrow 2}^P$:

$$\mathcal{I}_{x \rightarrow 2, y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ y & = 1 \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

' Rxy ' is true on this interpretation, since 2 bears the relation R to 1. Thus, ' Rxy ' is true on some y -variant of $\mathcal{I}_{x \rightarrow 2}^P$. So ' $(\exists y)Rxy$ ' is true on the interpretation $\mathcal{I}_{x \rightarrow 2}^P$.

Therefore, ' $(\exists y)Rxy$ ' is true on every x -variant of the interpretation \mathcal{I}^P . So ' $(\forall x)(\exists y)Rxy$ ' is true on the interpretation \mathcal{I}^P .

$(\forall x)(\exists y)Ryx$

Consider next the wff ' $(\forall x)(\exists y)Ryx$ '. This wff is true on the interpretation \mathcal{I}^P iff ' $(\exists y)Ryx$ ' is true on every x -variant of \mathcal{I}^P . As before, there are two x -variants of \mathcal{I}^P :

$$\mathcal{I}_{x \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 1 \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{x \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

Let's start with the second, $\mathcal{I}_{x \rightarrow 2}^P$. ' $(\exists y)Ryx$ ' is true on this interpretation iff ' Ryx ' is true on *some* y -variant of $\mathcal{I}_{x \rightarrow 2}^P$. There are two y -variants of $\mathcal{I}_{x \rightarrow 2}^P$:

$$\mathcal{I}_{x \rightarrow 2, y \rightarrow 1}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ y & = 1 \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases} \quad \text{and} \quad \mathcal{I}_{x \rightarrow 2, y \rightarrow 2}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ x & = 2 \\ y & = 2 \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

On the first— $\mathcal{I}_{x \rightarrow 2, y \rightarrow 1}^P$ —' Ryx ' is false, since 1 does not bear the relation R to 2. On the second— $\mathcal{I}_{x \rightarrow 2, y \rightarrow 2}^P$ —' Ryx ' is false, since 2 does not bear the relation R to itself.

Thus, ' Ryx ' is not true on any y -variant of $\mathcal{I}_{x \rightarrow 2}^P$. So ' $(\exists y)Ryx$ ' is not true on the interpretation $\mathcal{I}_{x \rightarrow 2}^P$. So ' $(\exists y)Ryx$ ' is not true on every x -variant of \mathcal{I}^P . Therefore, ' $(\forall x)(\exists y)Ryx$ ' is not true on the interpretation \mathcal{I}^P .

Therefore, on the interpretation \mathcal{I}^P , ' $(\forall x)(\exists y)Rxy$ ' is true, yet ' $(\forall x)(\exists y)Ryx$ ' is false. So these wffs are not *PL*-equivalent. So, they mean different things in *PL*.

4.6.3 Translating ' $(\forall x)(\exists y)Rxy$ ' and ' $(\forall x)(\exists y)Ryx$ '

' $(\forall x)(\exists y)Rxy$ ', translated into English, says that everything in the domain bears the relation R to something in the domain. ' $(\forall x)(\exists y)Ryx$ ', on the other hand, says that everything in the domain *is borne* the relation R by something in the domain. Because, in the interpretation above, everything does *bear* the relation R to something—1 bears it to 1 and 2 bears it to 1—this claim is true. However, in the interpretation above, it is false that everything in the domain *is borne* the relation R by something. Nothing bears the relation R to 2. So the claim that everything in the domain is borne R by something is false.

Consider the following partial interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \text{the set of all people} \\ Pxy & = \mathbf{x} \text{ is the biological parent of } \mathbf{y} \end{cases}$$

(In order to not worry about who the first people were, or whether they had parents, let's just suppose that there have been an infinite number of people and that every person has two biological parents, though not every person has children of their own.)

Then, ' $(\forall x)(\exists y)Pxy$ ' says that everybody is the parent of somebody. However, this is false, since not everybody has children. Those people don't have anybody that they are the parent of, so it is false that everybody is the parent of somebody.

$(\forall x)(\exists y)Pxy$, on the other hand, says that everybody has somebody who is their parent—that is, that everybody has a parent. And that is true. So $(\forall x)(\exists y)Pxy$ can be false while $(\forall x)(\exists y)Pyx$ is true. So, those wffs are not *PL*-equivalent.

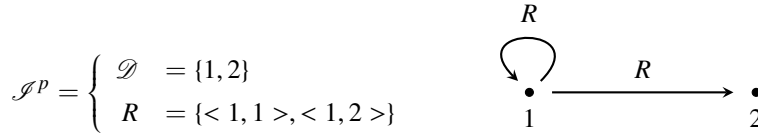
The lesson: changing the order of the bound variables can make a difference to the meaning and the truth-value of the wffs of *PL*.

4.6.4 Changing the Order of the Quantifiers and the Order of the Bound Variables

Suppose that, when we change the order of the quantifiers, we change the order of the bound variables as well. Then will we get a pair of wffs which are *PL*-equivalent? That is: are the following wffs of *PL* *PL*-equivalent?

$$\begin{aligned} &(\forall x)(\exists y)Rxy \\ &(\exists y)(\forall x)Ryx \end{aligned}$$

The answer is ‘no’. The first wff says that everything bears the relation *R* to something. The second wff says that something bears the relation *R* to everything. But either of these could be true without the other being true. Consider the following (partial) *PL*-interpretation.





On this interpretation, 2 does not bear the relation *R* to anything in the domain. So $(\forall x)(\exists y)Rxy$ is false, since not everything bears the relation *R* to something. That is: since ‘*Rxy*’ is false on both the variant interpretations $\mathcal{I}_{x \rightarrow 2, y \rightarrow 1}^P$ and $\mathcal{I}_{x \rightarrow 2, y \rightarrow 2}^P$, $(\exists y)Rxy$ is false on the *x*-variant interpretation $\mathcal{I}_{x \rightarrow 2}^P$. And if $(\exists y)Rxy$ is false on some *x*-variant of the interpretation \mathcal{I}^P , then $(\forall x)(\exists y)Rxy$ is false on \mathcal{I}^P .

Nevertheless, $(\exists y)(\forall x)Ryx$ is true, since something (namely, 1) bears the relation *R* to everything. That is: ‘*Ryx*’ is true on both $\mathcal{I}_{y \rightarrow 1, x \rightarrow 1}^P$ and $\mathcal{I}_{y \rightarrow 1, x \rightarrow 2}^P$. Therefore, $(\forall x)Ryx$ is true on $\mathcal{I}_{y \rightarrow 1}^P$. And if $(\forall x)Ryx$ is true on some *y*-variant of \mathcal{I}^P , then $(\exists y)(\forall x)Ryx$ is true on the interpretation \mathcal{I}^P .

To see that $(\forall x)(\exists y)Rxy$ can be true while $(\exists y)(\forall x)Ryx$ is false, consider the following interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ R & = \{ \langle 1, 1 \rangle, \langle 2, 2 \rangle \} \end{cases}$$





Here, $(\forall x)(\exists y)Rxy$ is true, since everything bears the relation R to something. 1 bears the relation R to itself, and 2 bears the relation R to itself. That is: Rxy is true on the interpretation $\mathcal{I}_{x \rightarrow 1, y \rightarrow 1}^P$, so $(\exists y)Rxy$ is true on the interpretation $\mathcal{I}_{x \rightarrow 1}^P$. And Rxy is true on the interpretation $\mathcal{I}_{x \rightarrow 2, y \rightarrow 2}^P$, so $(\exists y)Rxy$ is true on the interpretation $\mathcal{I}_{x \rightarrow 2}^P$. So $(\exists y)Rxy$ is true on every x -variant of the interpretation \mathcal{I}^P . So $(\forall x)(\exists y)Rxy$ is true on the interpretation \mathcal{I}^P .

Nevertheless, $(\exists y)(\forall x)Ryx$ is false on this interpretation. For Ryx is false on the interpretation $\mathcal{I}_{y \rightarrow 1, x \rightarrow 2}^P$, so $(\forall x)Ryx$ is false on the interpretation $\mathcal{I}_{y \rightarrow 1}^P$, and Ryx is false on the interpretation $\mathcal{I}_{y \rightarrow 2, x \rightarrow 1}^P$, so $(\forall x)Ryx$ is false on the interpretation $\mathcal{I}_{y \rightarrow 2}^P$. So $(\forall x)Ryx$ is false on every y -variant of the interpretation \mathcal{I}^P . So $(\exists y)(\forall x)Ryx$ is false on the interpretation \mathcal{I}^P .

Chapter 5

Predicate Logic Trees

5.1 Notation and Terminology

A bit of review: recall that we are using expressions like

$$\mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}], \mathbf{Q}[\mathbf{x} \rightarrow \mathbf{t}]$$

to refer to the wffs of *PL* that you get when you replace every *free* occurrence of ' \mathbf{x} ' in ' \mathbf{P} ' and ' \mathbf{Q} ' with the term ' \mathbf{t} '. That is: given a wff ' \mathbf{P} ', you get the wff ' $\mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}]$ ' by going through ' \mathbf{P} ', and every time ' \mathbf{x} ' appears free, you swap it out for the term ' \mathbf{t} '.

Also, a bit of terminology: If ' \mathbf{a} ' is a constant, then we will refer to ' $\mathbf{P}[\mathbf{x} \rightarrow \mathbf{a}]$ ' as a *substitution instance* of the quantified wffs ' $(\forall \mathbf{x})\mathbf{P}$ ' and ' $(\exists \mathbf{x})\mathbf{P}$ '.

Thus, ' $Pa \supset Qa$ ' is a *substitution instance* of the wff ' $(\forall z)(Pz \supset Qz)$ ' and ' $Pk \& (\forall y)S ky$ ' is a substitution instance of the wff ' $(\exists x)(Px \& (\forall y)S xy)$ '.

Here's a special case. Consider the wff

$$(\forall x)(\exists x)Px$$

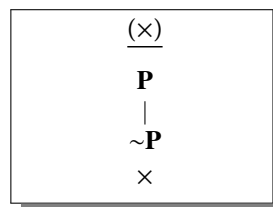
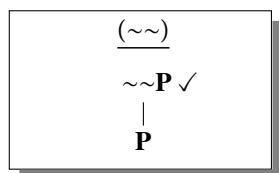
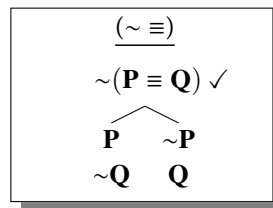
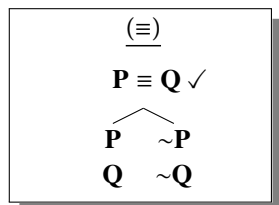
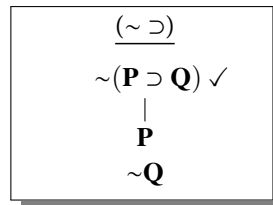
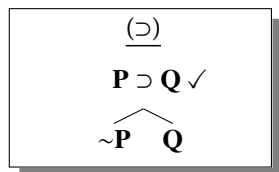
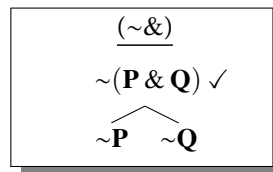
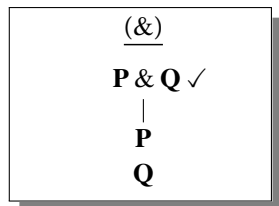
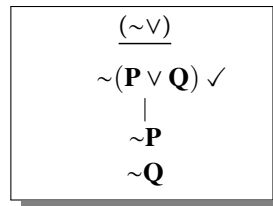
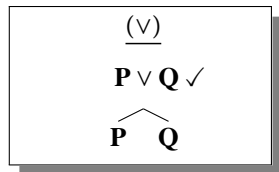
When we remove the universal quantifier ' $(\forall x)$ ', there are no free occurrences of ' x ' in the wff that we have left behind. So, we needn't replace any free occurrences of ' x ' with any constant. Therefore,

$$(\exists x)Px$$

Is a substitution instance of ' $(\forall x)(\exists x)Px$ '.

5.2 Predicate Logic Trees

We're now going to learn about a procedure for testing the consistency of sets of wffs of *PL*. This procedure is going to be very similar to the one that we learned about for *SL*. In fact, we will simply take all of the tree rules from *SL* and add to it four new rules. To refresh your memory, here are all of the rules from *SL*:



All of these rules may be applied to wffs of *PL* just as well as they were applied to wffs of *SL*. For instance, the following will be a correct *PL* tree:

$$\begin{array}{c}
 Pab \supset \sim Fj \quad \checkmark \\
 Fj \\
 Pab \\
 \swarrow \quad \searrow \\
 \sim Pab \quad \sim Fj \\
 \times \quad \times
 \end{array}$$

To these ten rules, we will add four more. The first rule we will learn about will be for wffs whose main operator is the universal quantifier, $\ulcorner (\forall \mathbf{x})\mathbf{P} \urcorner$.

5.2.1 Rule for Universally Quantified Wffs

Here is the rule for wffs of the form $\ulcorner (\forall \mathbf{x})\mathbf{P} \urcorner$:

$$\begin{array}{c}
 \underline{(\forall)} \\
 (\forall \mathbf{x})\mathbf{P} \quad \textcircled{\mathbf{t}} \\
 | \\
 \mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}]
 \end{array}$$

for any constant or free variable \mathbf{t}

Here's how to read this rule: if you have a wff of the form $\ulcorner (\forall \mathbf{x})\mathbf{P} \urcorner$ appearing at some point on an open branch, then you may place any constant $\ulcorner \mathbf{t} \urcorner$ in a circle next to the wff $\ulcorner (\forall \mathbf{x})\mathbf{P} \urcorner$, and write down the substitution instance $\ulcorner \mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}] \urcorner$ at the end of every open branch on which $\ulcorner (\forall \mathbf{x})\mathbf{P} \urcorner$ appears.

Note: you do not check off the universally quantified wff $\ulcorner (\forall \mathbf{x})\mathbf{P} \urcorner$. You may have to write down multiple substitution instances of this wff before you are done. Thus, you may end up writing many terms in circles next to the universally quantified wff.

For instance, consider the tree with the wffs $\ulcorner (\forall x)\sim Px \urcorner$ and $\ulcorner Pa \vee Pb \urcorner$ at its root:

$$\begin{array}{c}
 (\forall x)\sim Px \\
 Pa \vee Pb
 \end{array}$$

Let's begin with the disjunction $\ulcorner Pa \vee Pb \urcorner$. The rule for (\vee) tell us that we may place a check mark next to this wff, branch the tree, and write $\ulcorner Pa \urcorner$ on the left side and write $\ulcorner Pb \urcorner$ on the right.

$$\begin{array}{c}
 (\forall x)\sim Px \\
 Pa \vee Pb \quad \checkmark \\
 \wedge \\
 Pa \quad Pb
 \end{array}$$

Now, we only have the universally quantified wff to work with. We cannot check it off, but we can write down substitution instances of it. We could choose to instantiate any constant we like, but we should be judicious in our choice of constant. In particular, it makes sense to use the constants that already appear on the same branch as ‘ $(\forall x)\sim Px$ ’. Let’s start with ‘ a ’.

$$\begin{array}{c}
 (\forall x)\sim Px \quad (a) \\
 Pa \vee Pb \quad \checkmark \\
 \wedge \\
 Pa \quad Pb \\
 | \quad | \\
 \sim Pa \quad \sim Pa \\
 \times
 \end{array}$$

Since ‘ Pa ’ and ‘ $\sim Pa$ ’ appear on the same branch, that branch closes, by rule (\times).

Now, we’ve already applied a rule to every wff which is not an atomic wff or the negation of an atomic wff. Does that mean that we are done? *No*. Though we applied a rule to ‘ $(\forall x)\sim Px$ ’, we did not check it off. It may still have rules applied to it. We may additionally instantiate the constant ‘ b ’, as so:

$$\begin{array}{c}
 (\forall x)\sim Px \quad (a) \quad (b) \\
 Pa \vee Pb \quad \checkmark \\
 \wedge \\
 Pa \quad Pb \\
 | \quad | \\
 \sim Pa \quad \sim Pa \\
 \times \quad | \\
 \quad \quad \sim Pb \\
 \quad \quad \times
 \end{array}$$

Since ‘ Pb ’ and ‘ $\sim Pb$ ’ appear on the same branch, that branch closes.

This particular tree closes, but not every tree with a universally quantified wff will close. Consider, for instance, the tree with the wffs ‘ $(\forall x)(Px \supset Qx)$ ’ and ‘ Pa ’ at its root:

$$\begin{array}{c}
 (\forall x)(Px \supset Qx) \\
 Pa
 \end{array}$$

Since ‘ a ’ appears on this tree, it makes sense to begin by instantiating the constant ‘ a ’, as so:

$$\begin{array}{c}
 (\forall x)(Px \supset Qx) \textcircled{a} \\
 Pa \\
 | \\
 Pa \supset Qa
 \end{array}$$

We may then apply the rule for (\supset) to get:

$$\begin{array}{c}
 (\forall x)(Px \supset Qx) \textcircled{a} \\
 Pa \\
 | \\
 Pa \supset Qa \checkmark \\
 \swarrow \quad \searrow \\
 \sim Pa \quad Qa \\
 \times
 \end{array}$$

This tree hasn't yet closed, but we haven't checked off every wff, since we never check off a universally quantified wff. We could still, if we wished, apply the rule (\forall) to the wff ' $(\forall x)(Px \supset Qx)$ ' by instantiating a different constant, for instance, 'b':

$$\begin{array}{c}
 (\forall x)(Px \supset Qx) \textcircled{a} \textcircled{b} \\
 Pa \\
 | \\
 Pa \supset Qa \checkmark \\
 \swarrow \quad \searrow \\
 \sim Pa \quad Qa \\
 \times \quad | \\
 \quad Pb \supset Qb
 \end{array}$$

Since there are an infinite number of terms of PL , we could go on doing this forever. That means that we will have to alter our definition of what it is for a tree to remain open. Our new definition will say that, if we have applied the rule (\forall) by instantiating every constant or free variable which appears on an open branch with a universally quantified wff ' $(\forall \mathbf{x})\mathbf{P}$ ', then we are done. If a branch of the tree containing a universally quantified wff ' $(\forall \mathbf{x})\mathbf{P}$ ' remains open after instantiating every constant or free variable appearing on that branch, then the tree remains open.

To complete a tree:

1. Apply the relevant rules to all wffs appearing on open branches, in any order you like.
2. If a wff $\lceil \mathbf{P} \rceil$ and its negation $\lceil \sim \mathbf{P} \rceil$ appear on the same branch, then close that branch by writing '×' at the bottom of the branch.
3. If every branch closes, then you are done; in this case, we say that *the tree closes*.
4. If you have applied every relevant rule to every wff on every open branch which is not atomic or the negation of an atomic wff, **and, you have applied the rule (\forall) to every universally quantified wff appearing on an open branch by instantiating every constant or free variable which appears on an open branch with that universally quantified wff**, then you are done; if, after doing this, there remains an open branch, then we say that *the tree remains open*.

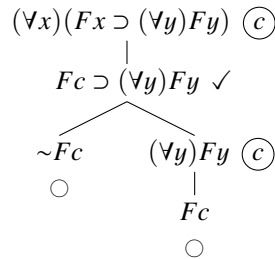
Note that we must still apply every relevant rule to every wff on the tree which is not atomic or a negation of an atomic wff. That means that, even if there are no constants or free variables appearing on an open branch with a wff of the form $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$, we must still apply the rule (\forall) to that wff by instantiating some *new* term in that wff. For instance, consider the tree beginning with $\lceil (\forall x)(Fx \supset (\forall y)Fy) \rceil$. There are no constants or free variables appearing in that wff. Nevertheless, we must still apply a rule to it, which means that we must pick some new term—say 'c'—and write down a substitution instance of $\lceil (\forall x)(Fx \supset (\forall y)Fy) \rceil$, as so:

$$\begin{array}{c} (\forall x)(Fx \supset (\forall y)Fy) \text{ (C)} \\ | \\ Fc \supset (\forall y)Fy \end{array}$$

We may now apply the rule for (\supset) as follows.

$$\begin{array}{c} (\forall x)(Fx \supset (\forall y)Fy) \text{ (C)} \\ | \\ Fc \supset (\forall y)Fy \checkmark \\ \swarrow \quad \searrow \\ \sim Fc \quad (\forall y)Fy \end{array}$$

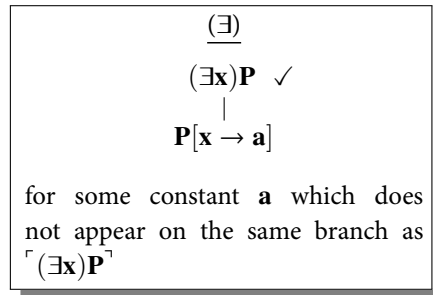
We must apply the rule (\forall) to $\lceil (\forall y)Fy \rceil$. However, since the constant 'c' appears on the same branch as $\lceil (\forall y)Fy \rceil$, we do not need to select a new instantiating constant. We may simply instantiate 'c', as follows.



At this point, we have applied the relevant rule to every wff appearing on every open branch of the tree, and, for every universally quantified wff appearing on an open branch of the tree and every constant or free variable appearing on that branch, we have instantiated that universally quantified wff with that constant or free variable. So, we are done with the tree. Since a branch remains open, the tree remains open.

5.2.2 Rule for Existentially Quantified Wffs

The rule for existentially quantified wffs—wffs of the form $\ulcorner (\exists \mathbf{x})\mathbf{P} \urcorner$ —is given below.



Here's how to read this rule: if you have a wff of the form $\ulcorner (\exists \mathbf{x})\mathbf{P} \urcorner$ appearing on an open branch of the tree, then you may, at any time you like, pick an *entirely new constant*, $\ulcorner \mathbf{a} \urcorner$ —one which does not appear on the same branch as $\ulcorner (\exists \mathbf{x})\mathbf{P} \urcorner$ —and write down the substitution instance $\mathbf{P}[\mathbf{x} \rightarrow \mathbf{a}]$.

For instance, consider a tree with the wffs $\ulcorner (\forall x)Fx \urcorner$ and $\ulcorner (\exists x)\sim Fx \urcorner$ at its root.

$$\begin{array}{c}
 (\forall x)Fx \\
 (\exists x)\sim Fx
 \end{array}$$

We may apply the rule (\exists) to the wff $\ulcorner (\exists x)\sim Fx \urcorner$ by checking it off and writing down a substitution instance of it. When we do so, we must choose *an entirely new constant—one which does not appear elsewhere on the branch*. Here, we could choose any constant, since no constants currently appear on the tree. Let's choose the constant 'j':

$$\begin{array}{c}
 (\forall x)Fx \\
 (\exists x)\sim Fx \quad \checkmark \\
 | \\
 \sim Fj
 \end{array}$$

Before we are done with the tree, we must also apply the rule (\forall) to ' $(\forall x)Fx$ ' by instantiating every constant or free variable which appears on the same branch as it. Now, ' j ' appears on the same branch as ' $(\forall x)Fx$ ', so we must instantiate the constant ' j ', as shown:

$$\begin{array}{c}
 (\forall x)Fx \quad (j) \\
 (\exists x)\sim Fx \quad \checkmark \\
 | \\
 \sim Fj \\
 | \\
 Fj \\
 \times
 \end{array}$$

Consider next the tree beginning with ' $(\forall x)(Fx \supset Gx)$ ' and ' $(\exists y)(Fy \& \sim Gy)$ ' at its root:

$$\begin{array}{c}
 (\forall x)(Fx \supset Gx) \\
 (\exists y)(Fy \& \sim Gy)
 \end{array}$$

We could start by instantiating the wff ' $(\forall x)(Fx \supset Gx)$ '. However, in general, it's a good idea to start by applying the rule (\exists) before applying the rule (\forall), so I'll do that here. I'll pick an entirely new constant—let's go with ' k '—and write down the substitution instance ' $Fk \& \sim Gk$ ' of ' $(\exists y)(Fy \& \sim Gy)$ '.

$$\begin{array}{c}
 (\forall x)(Fx \supset Gx) \\
 (\exists y)(Fy \& \sim Gy) \quad \checkmark \\
 | \\
 Fk \& \sim Gk
 \end{array}$$

Applying the rule ($\&$) gives us:

$$\begin{array}{c}
 (\forall x)(Fx \supset Gx) \\
 (\exists y)(Fy \& \sim Gy) \quad \checkmark \\
 | \\
 Fk \& \sim Gk \quad \checkmark \\
 | \\
 Fk \\
 \sim Gk
 \end{array}$$

Now, we need to apply the rule (\forall) to ' $(\forall x)(Fx \supset Gx)$ '. We could pick any constant to instantiate we like, but we won't be done with the tree until we instantiate the constant ' k ', so it makes sense to start with ' k ':

$$\begin{array}{c}
 (\forall x)(Fx \supset Gx) \text{ (k)} \\
 (\exists y)(Fy \ \& \ \sim Gy) \ \checkmark \\
 | \\
 Fk \ \& \ \sim Gk \ \checkmark \\
 | \\
 Fk \\
 \sim Gk \\
 | \\
 Fk \supset Gk
 \end{array}$$

When we apply the rule (\supset) to ' $Fk \supset Gk$ ', the tree closes:

$$\begin{array}{c}
 (\forall x)(Fx \supset Gx) \text{ (k)} \\
 (\exists y)(Fy \ \& \ \sim Gy) \ \checkmark \\
 | \\
 Fk \ \& \ \sim Gk \ \checkmark \\
 | \\
 Fk \\
 \sim Gk \\
 | \\
 Fk \supset Gk \ \checkmark \\
 \wedge \\
 \begin{array}{cc}
 \sim Fk & Gk \\
 \times & \times
 \end{array}
 \end{array}$$

5.2.3 Rules for Negations of Quantified Wffs

Suppose that we have a wff which is the negation of a universally or existentially quantified wff. That is, suppose that we have a wff of the form ' $\sim(\forall \mathbf{x})\mathbf{P}$ ' or ' $\sim(\exists \mathbf{x})\mathbf{P}$ '. Then, the following rules tell us that we may, at any time, place a check next to this wff, push the negation in past the quantifier, and swap the universal quantifier with an existential quantifier; or, alternatively, swap the existential quantifier with a universal quantifier.

$$\boxed{
 \begin{array}{c}
 \underline{(\sim\forall)} \\
 \sim(\forall \mathbf{x})\mathbf{P} \ \checkmark \\
 | \\
 (\exists \mathbf{x})\sim\mathbf{P}
 \end{array}
 }$$

$$\boxed{
 \begin{array}{c}
 \underline{(\sim\exists)} \\
 \sim(\exists \mathbf{x})\mathbf{P} \ \checkmark \\
 | \\
 (\forall \mathbf{x})\sim\mathbf{P}
 \end{array}
 }$$

For instance, consider the tree with the wffs ' $(\forall y)(Ty \supset Uy)$ ', ' $\sim(\forall x)Ux$ ', and ' $(\forall x)Tx$ ' at its root:

$$\begin{array}{c}
 (\forall y)(Ty \supset Uy) \\
 \sim(\forall z)Uz \\
 (\forall x)Tx
 \end{array}$$

We may apply the rule $(\sim\forall)$ to ' $\sim(\forall z)Uz$ ' by checking off ' $\sim(\forall z)Uz$ ' and writing down ' $(\exists z)\sim Uz$ ', as shown:

$$\begin{array}{c}
 (\forall y)(Ty \supset Uy) \\
 \sim(\forall z)Uz \quad \checkmark \\
 (\forall x)Tx \\
 | \\
 (\exists z)\sim Uz
 \end{array}$$

We may then apply the rule (\exists) to ' $(\exists z)\sim Uz$ ', as shown:

$$\begin{array}{c}
 (\forall y)(Ty \supset Uy) \\
 \sim(\forall z)Uz \quad \checkmark \\
 (\forall x)Tx \\
 | \\
 (\exists z)\sim Uz \quad \checkmark \\
 | \\
 \sim Uk
 \end{array}$$

Next, we may apply the rule (\forall) to ' $(\forall y)(Ty \supset Uy)$ ' by instantiating that universally quantified wff with the constant ' k ':

$$\begin{array}{c}
 (\forall y)(Ty \supset Uy) \quad (k) \\
 \sim(\forall z)Uz \quad \checkmark \\
 (\forall x)Tx \\
 | \\
 (\exists z)\sim Uz \quad \checkmark \\
 | \\
 \sim Uk \\
 | \\
 Tk \supset Uk
 \end{array}$$

And we may apply the rule (\forall) to ' $(\forall x)Tx$ ' by instantiating that universally quantified wff with the constant ' k ':

$$\begin{array}{c}
 (\forall y)(Ty \supset Uy) \text{ (k)} \\
 \sim(\forall z)Uz \checkmark \\
 (\forall x)Tx \text{ (k)} \\
 | \\
 (\exists z)\sim Uz \checkmark \\
 | \\
 \sim Uk \\
 | \\
 Tk \supset Uk \\
 | \\
 Tk
 \end{array}$$

When we apply the rule for (\supset), the tree closes.

$$\begin{array}{c}
 (\forall y)(Ty \supset Uy) \text{ (k)} \\
 \sim(\forall z)Uz \checkmark \\
 (\forall x)Tx \text{ (k)} \\
 | \\
 (\exists z)\sim Uz \checkmark \\
 | \\
 \sim Uk \\
 | \\
 Tk \supset Uk \checkmark \\
 | \\
 Tk \\
 \wedge \\
 \begin{array}{cc}
 \sim Tk & Uk \\
 \times & \times
 \end{array}
 \end{array}$$

5.3 Strategies for Applying Rules

As with *SL*, we may apply rules in any order; and, consequently, there are multiple correct trees for any given root. The following are some rough-and-ready guidelines to follow to keep your trees as tidy as possible:

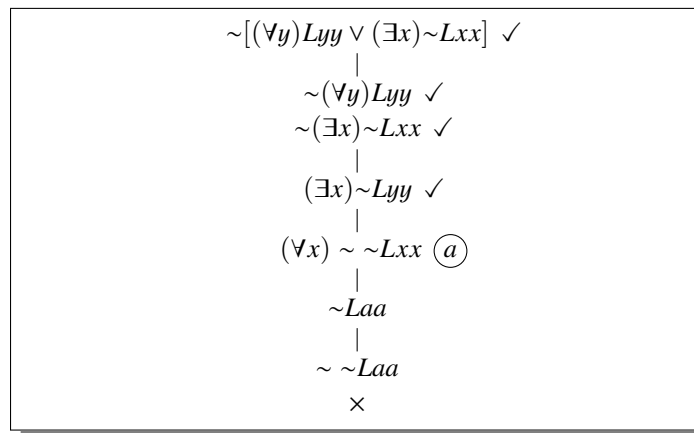
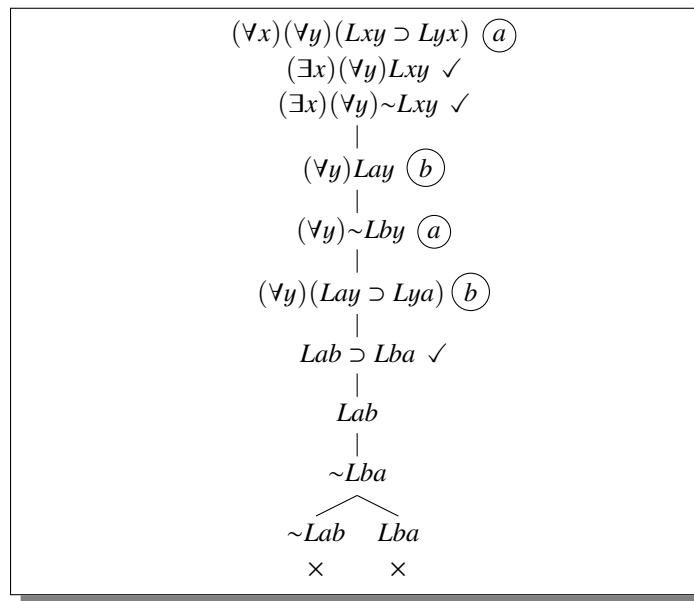
1. All else being equal, apply non-branching rules first.
2. All else being equal, apply rules to wffs which will lead to at least some branches closing before applying the rules to wffs which will not.
3. All else being equal, apply rules to wffs of the form $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ and $\lceil \sim(\forall \mathbf{x})\mathbf{P} \rceil$ before applying them to wffs of the form $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ and $\lceil \sim(\exists \mathbf{x})\mathbf{P} \rceil$.
4. All else being equal, when applying the rule (\forall), try to choose instantiating constants which will lead to branches closing before instantiating constants which

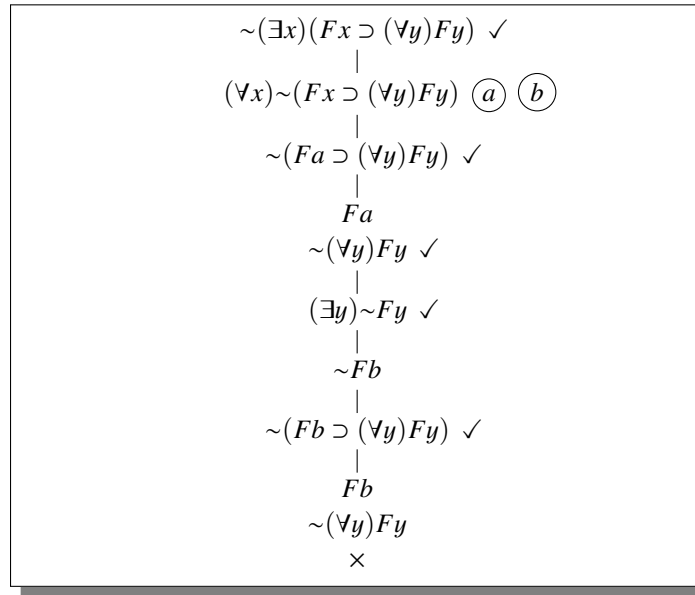
will leave all branches open.

5. All else being equal, work on longer wffs first.

5.4 Sample Predicate Logic Trees

Here are some sample *PL* trees:





5.5 Logical Properties of *PL*

Just as we did with *SL* trees, we can use *PL* trees to test for all the logical properties that we are interested in. Just to remind you, the family of logical properties we are interested in exploring are defined below and summarized in figure 6.4.

PL VALIDITY A *PL* argument is *PL* VALID iff there is no *PL* interpretation on which the premises are all true while the conclusion is false.

PL INVALIDITY A *PL* argument is *PL* INVALID iff there is some *PL* interpretation on which the premises are all true while the conclusion is false.

PL CONSISTENCY A set of wffs of *PL* $\{\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner\}$ is *PL* CONSISTENT iff there is some *PL* interpretation on which $\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner$ are all true.

PL INCONSISTENCY A set of wffs of *PL* $\{\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner\}$ is *PL* INCONSISTENT iff there is no *PL* interpretation on which $\ulcorner A_1 \urcorner, \ulcorner A_2 \urcorner, \dots, \ulcorner A_N \urcorner$ are all true.

PL EQUIVALENCE A pair of wffs of *PL*, $\ulcorner A \urcorner$ and $\ulcorner B \urcorner$, are *PL* EQUIVALENT iff $\ulcorner A \urcorner$ and $\ulcorner B \urcorner$ are true in all the same *PL* interpretations and false in all the same *PL* interpretations (*i.e.*, iff there is no *PL* interpretation on which $\ulcorner A \urcorner$ and $\ulcorner B \urcorner$ have different truth-values).

PL TAUTOLOGY A wff of *PL* $\ulcorner \mathbf{A} \urcorner$ is a *PL* TAUTOLOGY iff there is no *PL* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is false (*i.e.*, iff $\ulcorner \mathbf{A} \urcorner$ is true on every *PL* interpretation).

PL CONTRADICTION A wff of *PL* $\ulcorner \mathbf{A} \urcorner$ is a *PL* CONTRADICTION iff there is no *PL* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is true (*i.e.*, iff $\ulcorner \mathbf{A} \urcorner$ is false on every *PL* interpretation).

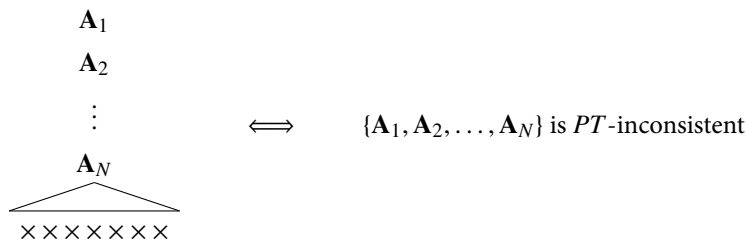
PL CONTINGENCY A wff of *PL* $\ulcorner \mathbf{A} \urcorner$ is a *PL* CONTINGENCY iff there is some *PL* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is true and some *PL* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is false.

Logical Property of <i>PL</i>	Applies Only To
<i>PL</i> Validity	<i>PL</i> arguments
<i>PL</i> Invalidity	<i>PL</i> arguments
<i>PL</i> Tautology	individual wffs of <i>PL</i>
<i>PL</i> Contradiction	individual wffs of <i>PL</i>
<i>PL</i> Contingency	individual wffs of <i>PL</i>
<i>PL</i> Equivalence	pairs of wffs of <i>PL</i>
<i>PL</i> Consistency	sets of wffs of <i>PL</i>
<i>PL</i> Inconsistency	sets of wffs of <i>PL</i>

Figure 5.1: Logical properties of *PL* and the kinds of things to which they apply.

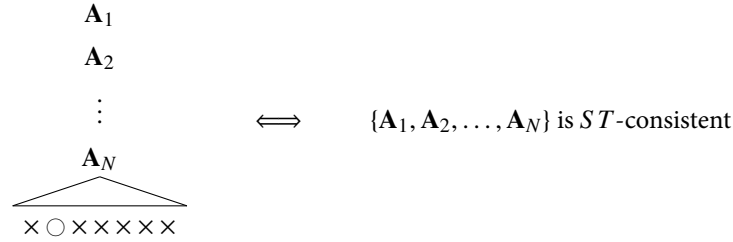
5.6 *PT*-Consistency

Suppose that you begin a tree with all and only the members of a set of wffs of *PL*, $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$, at its root, you apply the rules, and every branch of the tree closes (*i.e.*, the tree closes). Then, the set of sentences $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is *PT*-inconsistent.

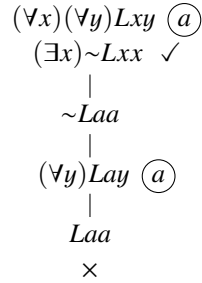


Suppose, on the other hand, that you begin a tree with all and only the members of a set of wffs of *PL*, $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$, at its root, you apply all of the required rules, and *not* every branch of the tree closes—at least one branch remains open (*i.e.*, the tree *doesn't*

close). Then, the set of wffs $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is *PT-consistent*.

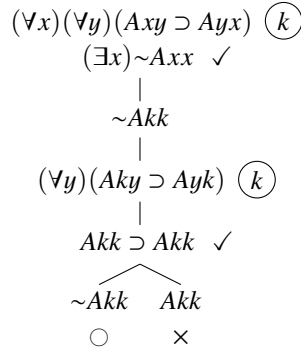


Is the set of sentences $\{(\forall x)(\forall y)Lxy, (\exists x)\sim Lxx\}$ *PT-consistent*?



Because the tree closes, we know that the set $\{(\forall x)(\forall y)Lxy, (\exists x)\sim Lxx\}$ is *PT-inconsistent*.

Is the set $\{(\forall x)(\forall y)(Axy \supset Ayx), (\exists x)\sim Axx\}$ *PT-consistent*?



Because the tree remains open, we know that the set $\{(\forall x)(\forall y)(Axy \supset Ayx), (\exists x)\sim Axx\}$ is *PT-consistent*.

It's important to note that the properties of *PT*-consistency and *PT*-inconsistency are defined very differently than the properties of *PL*-consistency and *PL*-inconsistency. The definition of *PL*-(in)consistency has to do with *PL* interpretations,

PL-CONSISTENCY A set of wffs of *PL*, $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is *PL-consistent* if and only if

there is some PL interpretation which makes each of $\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner$ true.

PL-INCONSISTENCY A set of wffs of PL , $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is PL -inconsistent if and only if there is no PL interpretation which makes each of $\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner$ true.

Whereas the definition of PT -(in)consistency has nothing to do with PL interpretations, and everything to do with whether certain trees close or not.

PT-CONSISTENCY A set of wffs of PL , $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is PT -consistent if and only if some completed PL tree which starts with $\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner$ remains open.

PT-INCONSISTENCY A set of wffs of PL , $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$ is PT -inconsistent if and only if some PT tree which starts with $\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner$ closes.

Nevertheless, there is an important relationship between PL -(in)consistency and PT -(in)consistency. While we will have to wait until later in the class to see the proof of this relationship, let me go ahead and inform you now that a set of wffs of PL is PL -consistent iff it is PT -consistent; and a set of wffs of PL is PL -inconsistent iff it is PT -inconsistent.

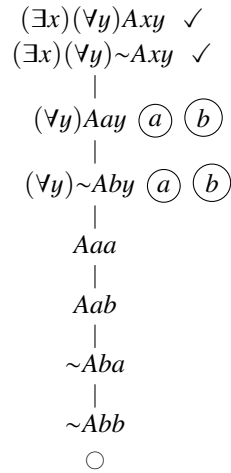
Fact: For any set of wffs of PL , $\{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N\}$, that set is PL -consistent if and only if it is PT -consistent, and PL -inconsistent if and only if it is PT -inconsistent.

This fact tells us that PT -(in)consistency is a property worth paying attention to. We can use the trees to tell us something about PL interpretations. If a tree closes, then we know that there is no PL interpretation that makes all the wffs at the root of the tree true. If the tree remains open, then we know that there is a PL interpretation which makes all of the wffs at the base of the tree true.

5.7 Reading Partial PL Interpretations off of Open Branches

Actually, we can use the trees to do more than this. We can use them not only to learn *that* there is a PL interpretation on which all of the wffs at the root of the tree are true. If a tree remains open, we may use the open branches of the tree to *read off* (partial) PL interpretations which make the wffs at the root of the tree true.

Here's an example to illustrate how we can do that: suppose that we start off with the set of wffs $\{(\exists x)(\forall y)Axy, (\exists x)(\forall y)\sim Axy\}$. Here is a completed tree for this set of wffs:



On this tree, there is one open branch. To read a partial *PL* interpretation off of this open branch, we begin by putting a thing in our domain for every constant or free variable which appears on that open branch. The only constants or free variables appearing on our open branch are a and b . So we let our domain contain these two things.

$$\mathcal{D} = \{a, b\}$$

Now: we look to the atomic wffs of *PL* which appear on the branch. Since Aaa appears on the open branch, we let a bear the relation A to itself. Since Aab appears on the open branch, we let a bear the relation A to b . Since no other atomic wffs appear on the open branch, we don't let anything else bear the relation A to anything else. Thus,

$$A = \{ \langle a, a \rangle, \langle a, b \rangle \}$$

And our completed partial interpretation looks like this:

$$\mathcal{I}^p = \begin{cases} \mathcal{D} & = \{a, b\} \\ A & = \{ \langle a, a \rangle, \langle a, b \rangle \} \end{cases}$$

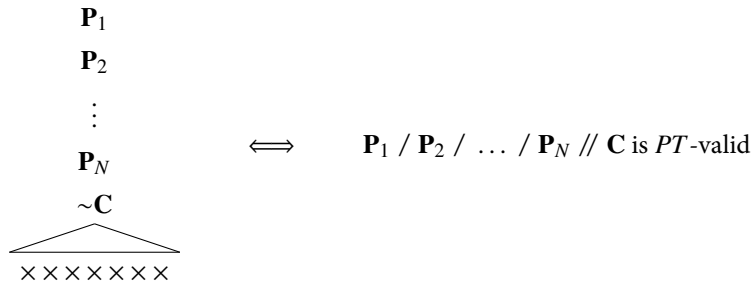
As you may confirm, on this partial interpretation, both ' $(\exists x)(\forall y)Axy$ ' and ' $(\exists x)(\forall y)\sim Axy$ ' are true.

5.8 *PT* Validity

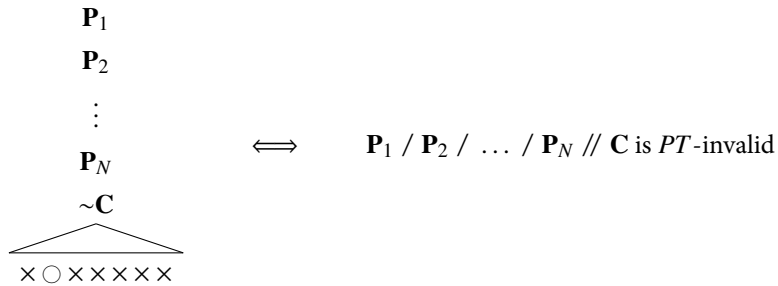
We have seen that an argument $\mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C}$ is valid if and only if the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim \mathbf{C}\}$ is inconsistent. Since the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N, \sim \mathbf{C}\}$ is *PL*-inconsistent if and only if it is *PT*-inconsistent, we may use truth-trees to tests arguments for *PL*-

validity.

Let's define a new notion: *PT*-validity. A *PL* argument is *PT*-valid if and only if some tree beginning with the wffs ' \mathbf{P}_1 ', ' \mathbf{P}_2 ', ..., ' \mathbf{P}_N ', and ' $\sim\mathbf{C}$ ' closes.



Similarly, we will say that a *PL*-argument is *PT*-invalid if and only if some completed tree beginning with the wffs ' \mathbf{P}_1 ', ' \mathbf{P}_2 ', ..., ' \mathbf{P}_N ', and ' $\sim\mathbf{C}$ ' remains open.



It's important to distinguish between *PL*-(in)validity and *PT*-(in)validity. The former is defined in terms of *PL* interpretations; whereas the latter is defined in terms of the trees. As it turns out, these two notions are deeply connected. Towards the end of the course, we will prove that an argument is *PL*-valid if and only if it is *PT*-valid, and that it is *PL*-invalid if and only if it is *PT*-invalid.

Fact: For any *PL*-argument $\mathbf{P}_1 / \mathbf{P}_2 / \dots / \mathbf{P}_N // \mathbf{C}$, that argument is *PL*-valid if and only if it is *PT*-valid, and *PL*-invalid if and only if it is *PT*-invalid.

However, this is something that we will have to prove. It is not obviously or straightforwardly true.

Consider the following *PL*-argument:

$$(\forall x)(Px \vee Qx) / (\forall x)(Px \supset Qx) // (\forall x)Qx$$

To test this argument for *PT*-validity, we begin by placing its premises and the negation of its conclusion at the top of a tree, as follows:

$$\begin{array}{l} (\forall x)(Px \vee Qx) \\ (\forall x)(Px \supset Qx) \\ \sim(\forall x)Qx \end{array}$$

When we complete the tree, we arrive at:

$$\begin{array}{c} (\forall x)(Px \vee Qx) \text{ (S)} \\ (\forall x)(Px \supset Qx) \text{ (S)} \\ \sim(\forall x)Qx \checkmark \\ | \\ (\exists x)\sim Qx \checkmark \\ | \\ \sim Qs \\ | \\ Ps \vee Qs \checkmark \\ | \\ Ps \supset Qs \checkmark \\ / \quad \backslash \\ \sim Ps \quad Qs \\ / \quad \backslash \quad \times \\ Ps \quad Qs \\ \times \quad \times \end{array}$$

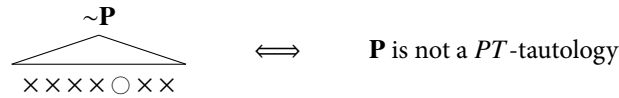
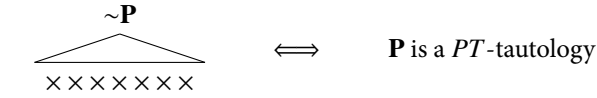
Since the tree closes, we know that the argument $(\forall x)(Px \vee Qx) / (\forall x)(Px \supset Qx) // (\forall x)Qx$ is *PT*-valid (and therefore, that it is *PL* valid).

Consider the *PL* argument

$$(\exists x)\sim Lxx // \sim(\forall x)(\forall y)(Lxy \supset \sim Lyx)$$

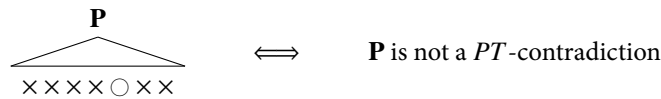
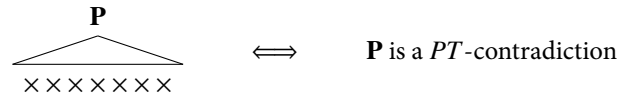
To see whether this argument is *PT*-valid, we complete the tree with its premise and the negation of its conclusion at the root. When we do so, we arrive at:

open.

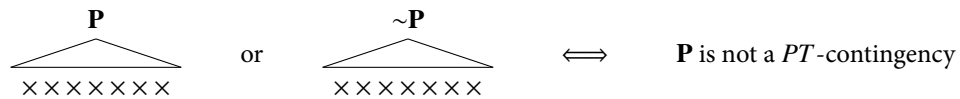
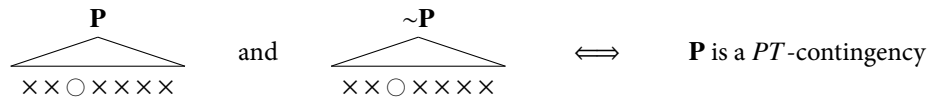


Here, again, it is important to keep the notion of a *PT*-tautology separate from the notion of a *PL*-tautology. The definition of *PL*-tautology has to do with *PL* interpretations, whereas the definition of *PT*-tautology has to do with whether a tree closes.

A wff of *PL* is a *PT*-contradiction if and only if some tree with $\lceil \mathbf{P} \rceil$ at its root closes. It is not a *PT*-contradiction if and only if some completed tree with $\lceil \mathbf{P} \rceil$ at its root remains open.



Finally, a wff of *PL* is a *PT*-contingency if and only if *both* some completed tree with $\lceil \mathbf{P} \rceil$ at its root remains open *and* some completed tree with $\lceil \sim \mathbf{P} \rceil$ at its root remains open. It is not a *PT*-contingency if and only if *either* some tree with $\lceil \mathbf{P} \rceil$ at its root closes *or* some tree with $\lceil \sim \mathbf{P} \rceil$ at its root closes.



For instance, consider the wff $\lceil (\exists x)(\forall y)(Fxx \supset Fyy) \rceil$. When we begin a tree with the negation of this wff at its root, the tree closes:

$$\begin{array}{c}
\sim(\exists x)(\forall y)(Fxx \supset Fyy) \checkmark \\
| \\
(\forall x)\sim(\forall y)(Fxx \supset Fyy) \text{ (a) (b)} \\
| \\
\sim(\forall y)(Faa \supset Fyy) \checkmark \\
| \\
(\exists y)\sim(Faa \supset Fyy) \checkmark \\
| \\
\sim(Faa \supset Fbb) \checkmark \\
| \\
Faa \\
\sim Fbb \\
| \\
\sim(\forall y)(Fbb \supset Fyy) \checkmark \\
| \\
(\exists y)\sim(Fbb \supset Fyy) \checkmark \\
| \\
\sim(Fbb \supset Fcc) \checkmark \\
| \\
Fbb \\
\sim Fcc \\
\times
\end{array}$$

So we know that ‘ $(\exists x)(\forall y)(Fxx \supset Fyy)$ ’ is a *PT*-tautology. (And that the wff ‘ $\sim(\exists x)(\forall y)(Fxx \supset Fyy)$ ’ is a *PT*-contradiction.)

Or consider the wff ‘ $(\exists x)(\exists y)Rxy$ ’. When we begin a tree with the negation of this wff at its root, the tree remains open:

$$\begin{array}{c}
\sim(\exists x)(\exists y)Rxy \checkmark \\
| \\
(\forall x)\sim(\exists y)Rxy \text{ (a)} \\
| \\
\sim(\exists y)Ray \checkmark \\
| \\
(\forall y)\sim Ray \text{ (a)} \\
| \\
\sim Raa \\
\circ
\end{array}$$

So we know that ‘ $(\exists x)(\exists y)Rxy$ ’ is not a *PT*-tautology. It is false on the partial *PL* interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{a\} \\ R & = \{\} \end{cases}$$

Similarly, when we begin a tree with ‘ $(\exists x)(\exists y)Rxy$ ’ at its root, the tree remains open:

$$\begin{array}{c} (\exists x)(\exists y)Rxy \checkmark \\ | \\ (\exists y)Ray \checkmark \\ | \\ Rab \end{array}$$

So we know that ‘ $(\exists x)(\exists y)Rxy$ ’ is not a *PT*-contradiction. It is true on the following partial *PL* interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{a, b\} \\ R & = \{ \langle a, b \rangle \} \end{cases}$$

And therefore, we know that ‘ $(\exists x)(\exists y)Rxy$ ’ is a *PT*-contingency.

5.10 *PT*-Equivalence

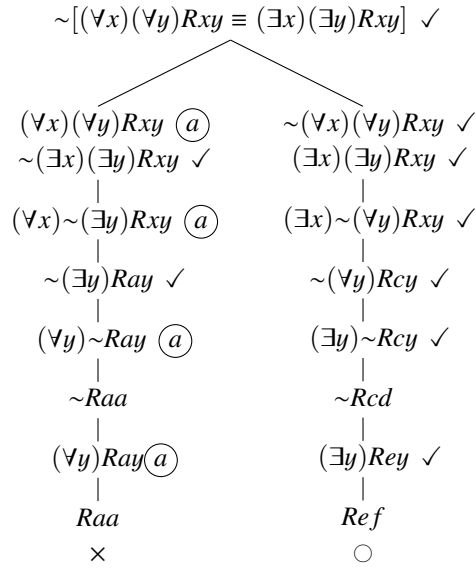
We may test for *PL*-equivalence using the tree method as follows: if some tree beginning with the wff ‘ $\sim(\mathbf{P} \equiv \mathbf{Q})$ ’ at its root closes, then ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are *PT*-equivalent.

$$\begin{array}{c} \sim(\mathbf{P} \equiv \mathbf{Q}) \\ \wedge \\ \times \times \times \times \times \times \end{array} \iff \mathbf{P} \text{ and } \mathbf{Q} \text{ are } PT\text{-equivalent}$$

If, on the other hand, some completed tree with ‘ $\sim(\mathbf{P} \equiv \mathbf{Q})$ ’ at its root remains open, then ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are not *PT*-equivalent.

$$\begin{array}{c} \sim(\mathbf{P} \equiv \mathbf{Q}) \\ \wedge \\ \times \times \times \times \circ \times \times \end{array} \iff \mathbf{P} \text{ and } \mathbf{Q} \text{ are not } PT\text{-equivalent}$$

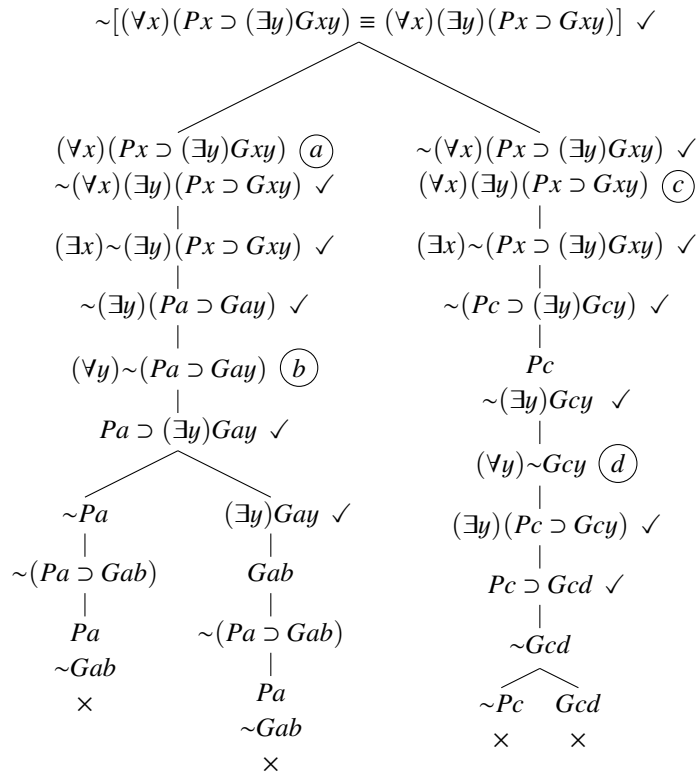
For instance, we may show that ‘ $(\forall x)(\forall y)Rxy$ ’ and ‘ $(\exists x)(\exists y)Rxy$ ’ are not *PT*-equivalent with the following tree:



' $(\exists x)(\exists y)Rxy$ ' is true and ' $(\forall x)(\forall y)Rxy$ ' is false on the following partial *PL* interpretation (read off the open branch):

$$\mathcal{I}^p = \begin{cases} \mathcal{D} & = \{c, d, e, f\} \\ R & = \{ \langle e, f \rangle \} \end{cases}$$

And we may show that ' $(\forall x)(Px \supset (\exists y)Gxy)$ ' and ' $(\forall x)(\exists y)(Px \supset Gxy)$ ' are *PT*-equivalent with the following tree:



5.11 Infinite Trees

Consider the tree which begins with the sole wff $(\forall x)(\exists y)Rxy$.

$$(\forall x)(\exists y)Rxy$$

We must begin this tree by applying the rule \forall , instantiating some constant—let's start with 'a':

$$\begin{array}{c} (\forall x)(\exists y)Rxy \text{ (a)} \\ | \\ (\exists y)Ray \end{array}$$

Then, we must apply the rule for \exists by writing down a substitution instance of $(\exists y)Ray$. However, we must choose an *entirely new name* to instantiate. We cannot use 'a' again. Let's choose 'b'.

$$\begin{array}{c}
 (\forall x)(\exists y)Rxy \quad (a) \\
 | \\
 (\exists y)Ray \quad \checkmark \\
 | \\
 Rab
 \end{array}$$

The tree hasn't closed; however, the tree is not yet complete, since we need to go back to $(\forall x)(\exists y)Rxy$ and write down a substitution instance with the constant 'b', since that constant now appears on the same branch as $(\forall x)(\exists y)Rxy$.

$$\begin{array}{c}
 (\forall x)(\exists y)Rxy \quad (a) \quad (b) \\
 | \\
 (\exists y)Ray \quad \checkmark \\
 | \\
 Rab \\
 | \\
 (\exists y)Rby
 \end{array}$$

And now we must write down a substitution instance of $(\exists y)Rby$. However, we cannot use the constants 'a' or 'b'. Both of them already appear on the same branch as $(\exists y)Rby$. It must be an entirely new name. Let's choose 'c':

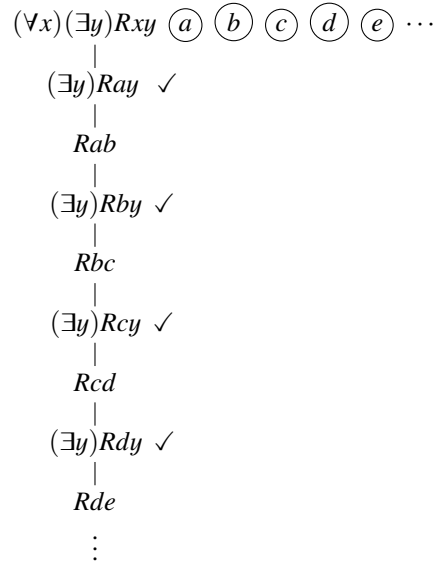
$$\begin{array}{c}
 (\forall x)(\exists y)Rxy \quad (a) \quad (b) \\
 | \\
 (\exists y)Ray \quad \checkmark \\
 | \\
 Rab \\
 | \\
 (\exists y)Rby \quad \checkmark \\
 | \\
 Rbc
 \end{array}$$

But now, we must go back to $(\forall x)(\exists y)Rxy$, and write down a substitution instance of it with the constant 'c', since that constant now appears on the same branch as it.

$$\begin{array}{c}
 (\forall x)(\exists y)Rxy \quad (a) \quad (b) \quad (c) \\
 | \\
 (\exists y)Ray \quad \checkmark \\
 | \\
 Rab \\
 | \\
 (\exists y)Rby \quad \checkmark \\
 | \\
 Rbc \\
 | \\
 (\exists y)Rcy
 \end{array}$$

And we will simply keep going in this way: introducing new names and then instanti-

ating them, over and over again, without end.



The completed tree is *infinitely long*. How can we tell? Well, we've established a pattern of the application of rules, and we can see that that pattern, if we continue on with it, will never terminate. (We could be more rigorous about this, but let's not.)

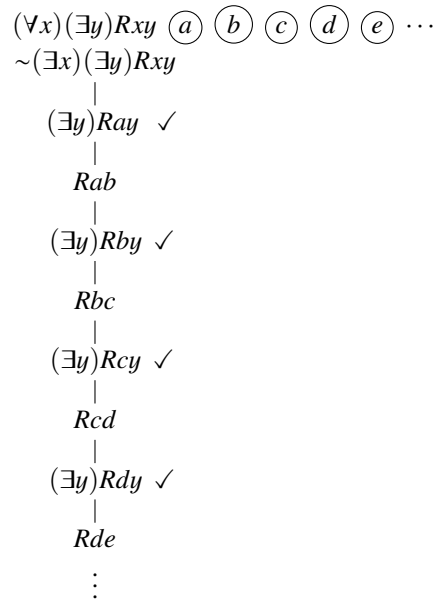
At no finite point will all of the required rules be applied. And at no finite point will the tree close. So...is $(\forall x)(\exists y)Rxy$ a *PT*-contradiction, or not? To answer that question, we must answer two others: does the infinite tree close, and is the infinite tree complete? If the tree goes on forever, then it looks like the tree must never close. (We will prove this rigorously later on in the course.) And if the tree never closes, then the tree must remain open. Ok. So the tree remains open. But is the tree complete? Well, at no finite point are all of the required rules applied. But the tree is infinite; so perhaps, in the infinite tree, all of the required rules are applied, even though, at no finite stage of construction are all of the required rules applied.

What we realized in the preceding paragraph is that it's important to distinguish two kinds of infinite trees: trees which are infinitely long and *complete* and those which are infinitely long but *incomplete*. Here's a nice way to think about these infinite trees: suppose that, at 11:00, we apply the first rule; at 11:30, we apply the second rule; at 11:40, we apply the third rule; at 11:45, we apply the fourth rule; at 11:48, we apply the fifth rule; at 11:50, we apply the sixth rule. In general, we apply the n th rule at $60/n$ minutes before 12:00. Then, when 12:00 rolls around, we will have applied an infinite number of rules. We will then have the infinite tree stretched out before us. And we can ask ourselves, of this infinite tree: is the tree complete?

The tree will be complete iff every wff which is neither atomic, the negation of an atomic, or universally quantified has been checked off, and, for every universally quantified wff appearing on the tree, we have instantiated every constant appearing on an open branch with that wff. If, once the infinite tree is complete, there are any constants on an open branch of the tree which are not instantiated in a universally quantified wff on the same branch, or if there are any wffs which are not checked off except for the universally quantified wffs, the atomic wffs, and the negations of the atomic wffs, then the infinite tree is not complete.

In the tree above, if we carry on introducing new names and then instantiating them forever, then every constant of *PL* will appear at some point on the tree, and every constant of *PL* will appear in a substitution instance of ‘ $(\forall x)(\exists y)Rxy$ ’. Every non-universally quantified wff will be checked off. So we can tell that the infinite tree that we will construct if we carry on in the same pattern will be a *complete* infinite tree.

On the other hand, consider the following infinite tree:



This tree is exactly like the one above, except that we have an additional wff at the root of the tree: ‘ $\sim(\exists x)(\exists y)Rxy$ ’. In the tree above, we establish the same pattern, and continue on in constructing the infinite tree exactly as we did above. In this case, when the infinite tree is written down, it will remain open; however, it will *not* be complete. That’s because the infinite tree will have a wff which is not an atomic, a negation of an atomic, or universally quantified, and which is not checked off—that is, the wff ‘ $\sim(\exists x)(\exists y)Rxy$ ’.

It is important to distinguish between infinite complete trees and infinite incomplete

trees because, if we weren't sensitive to that distinction, then we might say that the set $\{(\forall x)(\exists y)Rxy, \sim(\exists x)(\exists y)Rxy\}$ is *PT*-consistent; which is bad, because, that set is *PL*-inconsistent.

We avoid this result by saying that a set of wffs is *PT*-consistent iff some *completed* tree remains open; and *PT*-inconsistent iff every *completed* tree closes. But the above infinite tree is not completed; whereas the finite tree below is:

$$\begin{array}{c}
 (\forall x)(\exists y)Rxy \text{ (a)} \\
 \sim(\exists x)(\exists y)Rxy \checkmark \\
 | \\
 (\forall x)\sim(\exists y)Rxy \text{ (a)} \\
 | \\
 (\exists y)Ray \\
 | \\
 \sim(\exists y)Ray \\
 \times
 \end{array}$$

So we can conclude that the set $\{(\forall x)(\exists y)Rxy, \sim(\exists x)(\exists y)Rxy\}$ is *PT*-inconsistent. That's good, since that set is *PL*-inconsistent.

To sum up: some trees remain open once they are completed; but they require an infinite number of rules to be applied before they are complete. For those trees, you needn't complete the infinite tree; however, you must establish a pattern of applying rules which is such that, if that pattern of applying rules is carried out an infinite number of times, then:

1. The tree will remain open;
2. On every open branch, every wff which is neither an atomic wff, the negation of an atomic wff, nor a universally quantified wff will be checked off; and
3. For every universally quantified wff $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ appearing on an open branch, and every constant or free variable $\lceil \mathbf{t} \rceil$ appearing on that open branch, the rule \forall will have been applied to that wff with that constant or free variable by writing down the substitution instance $\lceil \mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}] \rceil$ at some point on that open branch.

For instance, consider the wff $(\exists x)(Fx \vee (\forall y)Fy)$. To test whether this wff is a *PT*-tautology, we construct the tree beginning with $\sim(\exists x)(Fx \vee (\forall y)Fy)$. That tree will be infinite:

$$\begin{array}{c}
\sim(\exists x)(Fx \vee (\forall y)Fy) \checkmark \\
| \\
(\forall x)\sim(Fx \vee (\forall y)Fy) \textcircled{a} \textcircled{b} \textcircled{c} \textcircled{d} \dots \\
| \\
\sim(Fa \vee (\forall y)Fy) \checkmark \\
| \\
\sim Fa \\
| \\
\sim(\forall y)Fy \checkmark \\
| \\
(\exists y)\sim Fy \checkmark \\
| \\
\sim Fb \\
| \\
\sim(Fb \vee (\forall y)Fy) \checkmark \\
| \\
\sim Fb \\
| \\
\sim(\forall y)Fy \checkmark \\
| \\
(\exists y)\sim Fy \checkmark \\
| \\
\sim Fc \\
| \\
\sim(Fc \vee (\forall y)Fy) \checkmark \\
| \\
\sim Fc \\
| \\
\sim(\forall y)Fy \checkmark \\
| \\
(\exists y)\sim Fy \checkmark \\
| \\
\sim Fd \\
| \\
\times
\end{array}$$

Now, we can see that we will continue on in this manner, writing down, for every constant of PL $\ulcorner \mathbf{a} \urcorner$ the following sequence:

$$\begin{array}{c}
\sim(F\mathbf{a} \vee (\forall y)Fy) \checkmark \\
| \\
\sim F\mathbf{a} \\
| \\
\sim(\forall y)Fy \checkmark \\
| \\
(\exists y)\sim Fy \checkmark \\
| \\
\sim F\mathbf{b}
\end{array}$$

where $\ulcorner \mathbf{b} \urcorner$ is some new constant of PL that hasn't yet appeared on the tree. As we go, we will check off every wff of PL that needs to be checked off, and every time we write down a new constant of PL $\ulcorner \mathbf{b} \urcorner$, we will write down a new substitution instance of

' $(\forall x)\sim(Fx \vee (\forall y)Fy)$ ', $\lceil \sim(F\mathbf{b} \vee (\forall y)Fy) \rceil$. So the tree will be infinitely long, but it will be complete, and it will never close. So the tree remains open. So we may conclude that ' $(\exists x)(Fx \vee (\forall y)Fy)$ ' is not a *PT*-tautology. It is false on the following interpretation (read off of the open branch):

$$\mathcal{I}^p = \begin{cases} \mathcal{D} & = \{a, b, c, d, e, \dots\} \\ F & = \{\} \end{cases}$$

Chapter 6

Predicate Logic with Identity and Functions

6.1 The Language *PLI*

6.1.1 Preliminary Orientation

We're going to enrich our logical language further. In the first place, we're going to introduce a special relation symbol: '='. The interpretation of this relation will be fixed. In *every* *PLI* interpretation, ' $t_1 = t_2$ ' will mean 'the thing denoted by t_1 is identical to the thing denoted by t_2 ', for all terms ' t_1 ' and ' t_2 ' of *PLI*. So, for instance, if we have the partial *PLI* interpretation

$$\mathcal{I}^P = \left\{ \begin{array}{l} \mathcal{D} = \text{the set of all people} \\ m = \text{Mark Twain} \\ s = \text{Samuel Clemens} \\ l = \text{David Lewis} \end{array} \right.$$

Then the wff

$$m = s$$

will say that Mark Twain is identical to (is the same person as) Samuel Clemens. And

$$\sim m = l$$

will say that Mark Twain is not identical to (is not the same person as) David Lewis. (Informally, we will write ‘ $\sim m = l$ ’ as ‘ $m \neq l$ ’.)

In the second place, we’re going to introduce a new kind of syntactic entity, known as a *function*. Given a term ‘ t ’ of *PLI*, and a function ‘ f ’ of *PLI*, ‘ $f(t)$ ’—read as ‘ f of t ’—will denote something in the domain of our interpretation, and will be treated syntactically just like any other term. So, for instance, given the partial *PLI* interpretation

$$\mathcal{I}^p = \left\{ \begin{array}{l} \mathcal{D} = \{1, 2, 3, 4, 5, \dots\} \\ a = 1 \\ f(\mathbf{x}) = \mathbf{x} + 1 \\ P\mathbf{x} = \mathbf{x} \text{ is prime} \end{array} \right.$$

‘ $f(a)$ ’ refers to 2 (since $1 + 1 = 2$). We may then write

$$Pf(a)$$

which says that $1 + 1$ is prime (which is true).

That’s the informal characterization. Now we’ll have to delve into the nitty-gritty.

6.1.2 Syntax for *PLI*

In this section, I’m going to tell you what the vocabulary of *PLI* is, and I’m going to tell you which expressions of *PLI* are grammatical—which are *well-formed formulae*, or ‘wffs’—just as I did for *PL*.

Vocabulary for *PLI*

The vocabulary of *PLI* include the following symbols:

1. for each $n \geq 1$, an infinite number of n -place predicates (any capital letter, along

with a superscript n —perhaps with subscripts)

$$\begin{array}{cccccccc} A^1 & B^1 & \dots & Z^1 & A_1^1 & \dots & Z_1^1 & A_2^1 & \dots \\ A^2 & B^2 & \dots & Z^2 & A_1^2 & \dots & Z_1^2 & A_2^2 & \dots \\ \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \dots \\ A^n & B^n & \dots & Z^n & A_1^n & \dots & Z_1^n & A_2^n & \dots \\ \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \dots \end{array}$$

2. for each $n \geq 1$, an infinite number of n -place function symbols (lowercase f , g , or h along with a superscript n —perhaps with subscripts)

$$\begin{array}{cccccccc} f^1 & g^1 & h^1 & f_1^1 & g_1^1 & h_1^1 & f_2^1 & \dots \\ f^2 & g^2 & h^2 & f_1^2 & g_1^2 & h_1^2 & f_2^2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots \\ f^n & g^n & h^n & f_1^n & g_1^n & h_1^n & f_2^n & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots \end{array}$$

3. An infinite number of *constants* (any lowercase letter between a and v , except for f , g , and h —perhaps with subscripts)

$$a, b, \dots, e, i, \dots, u, v, a_1, b_1, \dots, e_1, i_1, \dots, u_1, v_1, a_2, b_2, \dots$$

4. An infinite number of *variables* (lowercase w , x , y , or z —perhaps with subscripts)

$$w, x, y, z, w_1, x_1, y_1, z_1, w_2, x_2, \dots$$

5. Logical operators

$$\sim, \vee, \&, \supset, \equiv, \exists, \forall$$

6. The identity relation,

$$=$$

7. parentheses

$$(,)$$

Nothing else is included in the vocabulary of *PL*.

Grammar

Any sequence of the symbols in the vocabulary of *PLI* is a *formula* of *PLI*. For instance, all of the following are formulae of *PLI*:

$$\begin{aligned} V^{2800} x f^4 5 &==== \sim ((\supset \supset a n f) v) \\ P^1 = Q^2 = R^3 = S^4 = T^5 &==== \sim \sim \\ (\exists y)((\forall x) R^2 g(x, x) h(x) \supset \sim F^1 f^2(y, y)) \\ N^{54} x y \vee \sim \sim (\exists x) B^2 = x \end{aligned}$$

However, only one—the third—is a *well-formed formula* (or ‘wff’) of *PLI*. We specify what it is for a string of symbols from the vocabulary of *PLI* to be a wff of *PLI* with the following rules.

- f) If ‘ f^n ’ is an n -place function symbol and ‘ t_1 ’, ‘ t_2 ’, ..., ‘ t_n ’ are n terms, then ‘ $f^n(t_1, t_2, \dots, t_n)$ ’ is a term.
- \mathcal{F}) If ‘ \mathcal{F}^n ’ is an n -place predicate and ‘ t_1 ’, ‘ t_2 ’, ..., ‘ t_n ’ are n terms, then ‘ $\mathcal{F}^n t_1 t_2 \dots t_n$ ’ is a wff.
- \Rightarrow) If ‘ t_1 ’ and ‘ t_2 ’ are terms, then ‘ $t_1 = t_2$ ’ is a wff.
- \sim) If ‘ \mathbf{P} ’ is a wff, then ‘ $\sim \mathbf{P}$ ’ is a wff.
- $\&$) If ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are wffs, then ‘ $(\mathbf{P} \& \mathbf{Q})$ ’ is a wff.
- \vee) If ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are wffs, then ‘ $(\mathbf{P} \vee \mathbf{Q})$ ’ is a wff.
- \supset) If ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are wffs, then ‘ $(\mathbf{P} \supset \mathbf{Q})$ ’ is a wff.
- \equiv) If ‘ \mathbf{P} ’ and ‘ \mathbf{Q} ’ are wffs, then ‘ $(\mathbf{P} \equiv \mathbf{Q})$ ’ is a wff.
- \forall) If ‘ \mathbf{P} ’ is a wff and ‘ \mathbf{x} ’ is a variable, then ‘ $(\forall \mathbf{x})\mathbf{P}$ ’ is a wff.
- \exists) If ‘ \mathbf{P} ’ is a wff and ‘ \mathbf{x} ’ is a variable, then ‘ $(\exists \mathbf{x})\mathbf{P}$ ’ is a wff.
- Nothing else is a wff.

Note: none of ‘ f ’, ‘ \mathcal{F} ’, ‘ t ’, ‘ x ’, ‘ \mathbf{P} ’, and ‘ \mathbf{Q} ’ appear in the vocabulary of *PLI*. They are not *themselves* part of the vocabulary of *PLI*. Rather, we are using them here as VARIABLES ranging over the vocabulary of *PLI*.

All and only the strings of symbols that can be constructed by repeated application of the rules above are well-formed formulae. For instance, if we wanted to show that ‘ $((\forall x)R^2f^2(x, x)g^1(x) \supset \sim a = a)$ ’ is a wff of *PLI*, we could walk through the following steps to build it up:

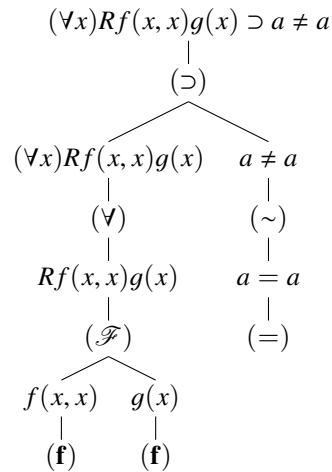
- a) ‘ $f^2(x, x)$ ’ is a term of *PLI* [from (f)]
- b) ‘ $g^1(x)$ ’ is a term of *PLI* [from (f)]
- c) So, ‘ $R^2f^2(x, x)g^1(x)$ ’ is a wff of *PLI* [from (a), (b), and (\mathcal{F})]
- d) So, ‘ $(\forall x)R^2f^2(x, x)g^1(x)$ ’ is a wff of *PLI* [from (c) and (\forall)]
- e) ‘ $a = a$ ’ is a wff of *PLI* [from (=)]
- f) So, ‘ $\sim a = a$ ’ is a wff of *PLI* [from (e) and (\sim)]
- g) So, ‘ $((\forall x)R^2f^2(x, x)g^1(x) \supset \sim a = a)$ ’ is a wff of *PLI* [from (d), (f), and (\supset)]

As before, we will adopt the convention of dropping the outermost parentheses in a wff of *PL* and dropping the superscripts on the predicates of *PLI*. We will *additionally* adopt the conventions of dropping the superscripts on *function* symbols of *PLI*. And we will adopt the convention of writing ‘ $\mathbf{t}_1 \neq \mathbf{t}_2$ ’ rather than ‘ $\sim \mathbf{t}_1 = \mathbf{t}_2$ ’. So, abiding by our informal conventions, we would write the wff of *PLI* ‘ $((\forall x)R^2f^2(x, x)g^1(x) \supset \sim a = a)$ ’ as:

$$(\forall x)Rf(x, x)g(x) \supset a \neq a$$

I’ll adopt these conventions from here on out.

We could, just as before, use SYNTAX TREES to represent the way that a wff of *PLI* is built up according to the rules for wffs given above. For instance, we could notate the proof given above as follows:



It is important to note, though, that not every sequence of symbols which appears on this syntax tree is itself a wff of *PLI*. ‘ $f(x, x)$ ’ and ‘ $g(x)$ ’ are *not* wffs of *PLI*. They are, rather, *terms* of *PLI*. ‘ $Rf(x, x)g(x)$ ’ is the first wff of *PLI* to appear on the left-most branch of this syntax tree (working our way up from the bottom).

6.1.3 Semantics for *PLI*

Just as in *PL*, we will provide the semantics for our language *PLI* by appealing to *interpretations*. A *PL* interpretation was defined as follows:

A *PL*-INTERPRETATION, \mathcal{I} , provides

1. A specification of which things fall in the *domain*, \mathcal{D} , of the interpretation.^a
2. For every variable of *PL*, a specification of which thing in the domain \mathcal{D} it represents.
3. For every constant of *PL*, a specification of which thing in the domain \mathcal{D} it represents.
4. For every predicate of *PL*, a specification of the property or relation it represents.

^a Note: the domain must be non-empty.

We will give a slightly altered definition of an interpretation for *PLI*:

A *PLI*-INTERPRETATION, \mathcal{I} , provides

1. A specification of which things fall in the *domain*, \mathcal{D} , of the interpretation.^a
2. For every variable of *PLI*, a specification of which thing in the domain \mathcal{D} it represents.
3. For every constant of *PLI*, a specification of which thing in the domain \mathcal{D} it represents.
4. For every predicate of *PLI*, a specification of the property or relation it represents.
5. For every function symbol of *PLI*, a specification of the function it represents. (Note: the function must be *total* on the domain \mathcal{D} .)

^a Note: the domain must be non-empty.

Just as with *PL*, we won't be able to specify a *full PLI* interpretation. So, instead, we will make do with *partial PLI* interpretations. A *partial PLI* interpretation is defined below.

Given a wff, set of wffs, or argument of *PLI*, a *PARTIAL PLI*-INTERPRETATION, \mathcal{I}^p provides:

1. A specification of which things fall in the *domain*, \mathcal{D} , of the partial interpretation.^a
2. For the *free* variables appearing in the wff, set of wffs, or argument of *PLI*, a specification of which thing in the domain \mathcal{D} they represent.
3. For the constants appearing in the wff, set of wffs, or argument of *PLI*, a specification of which thing in the domain \mathcal{D} they represent.
4. For the predicates appearing in the wff, set of wffs, or argument of *PLI*, a specification of the property or relation they represent.
5. For the function symbols appearing in the wff, set of wffs, or argument of *PLI*, a specification of the function they represent. (Note: the function must be *total* on the domain \mathcal{D} .)

^a Note: the domain must be non-empty.

(For now, don't worry about the requirement that the function be total—we'll come back to that requirement in a bit.)

For instance, suppose that we have the following wff of *PL*,

$$(\forall x)Lxf(x)$$

Here is a *partial interpretation* of this wff:

$$\mathcal{I}^p = \begin{cases} \mathcal{D} & = \{ \text{Adam, Betsy, Carol} \} \\ L\mathbf{x}\mathbf{y} & = \mathbf{x} \text{ loves } \mathbf{y} \\ f(\mathbf{x}) & = \text{the best friend of } \mathbf{x} \end{cases}$$

We specified the domain, \mathcal{D} . Since all of the variables are bound, we do not need to say which thing they refer to. There is just one predicate in this wff: a 2-place predicate, ' L '; and just one functions symbol: the 1-place function symbol, ' f '. We said that $L\mathbf{x}\mathbf{y}$ referred to the relation ' \mathbf{x} loves \mathbf{y} ' and we said that $f(\mathbf{x})$ refers to the best friend of \mathbf{x} . So we've provided a partial *PL* interpretation for this wff.

6.1.4 More on *PLI*-Interpretations of Function Symbols

Above, I interpreted ' f ' and ' L ' by just saying that ' $L\mathbf{x}\mathbf{y}$ ' refers to the relation ' \mathbf{x} loves \mathbf{y} ' and saying that ' $f(\mathbf{x})$ ' refers to the best friend of \mathbf{x} . This is fine, as far as it goes, except that you might not know who Adam loves, or who Betsy's best friend is. We saw before that we may specify the meaning of a relation with a set of ordered pairs, as in

$$L = \{ \langle \text{Adam, Adam} \rangle, \langle \text{Adam, Carol} \rangle, \langle \text{Betsy, Carol} \rangle \}$$

This tells us that Adam loves himself and Carol, Betsy loves Carol, Adam doesn't love Betsy, Betsy doesn't love herself or Adam, and Carol doesn't love anybody.

Similarly, there is another way of specifying the meaning of a function symbol of *PL* that will tell us precisely which objects in the domain are referred to by the expression ' $f(\mathbf{x})$ ', for every \mathbf{x} . Suppose that Adam's best friend is Betsy, Betsy's best friend is Carol, and Carol's best friend is Adam. Then, we may specify this explicitly by writing the meaning of ' f ' as follows:

$$f = \{ \langle \text{Adam, Betsy} \rangle, \langle \text{Betsy, Carol} \rangle, \langle \text{Carol, Carol} \rangle \}$$

This tells us that Adam's best friend is Betsy, Betsy's best friend is Carol, and Carol's best friend is herself.

So, both relations and functions get represented with a set of ordered pairs. What, then, is the difference between a function and a relation? The difference is this: while, for relations, a single thing may be related to multiple other things; for a function, each thing must be associated with *at most one* other thing.

Relations A RELATION on the domain \mathcal{D} is any set of ordered pairs $\langle a, b \rangle$ such that both a and b are in \mathcal{D} .

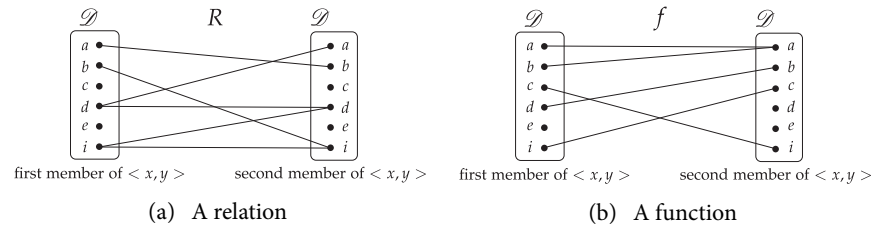


Figure 6.1: Figure 6.1a shows a relation on the domain \mathcal{D} . This relation is given by the set of ordered pairs $\{\langle a, b \rangle, \langle b, i \rangle, \langle d, a \rangle, \langle d, d \rangle, \langle i, d \rangle, \langle i, i \rangle\}$. Figure 6.1b shows a function on the domain \mathcal{D} . This function is given by the set of ordered pairs $\{\langle a, a \rangle, \langle b, a \rangle, \langle c, i \rangle, \langle d, b \rangle, \langle i, c \rangle\}$.

Functions A FUNCTION, f , on the domain \mathcal{D} is any set of ordered pairs $\langle a, b \rangle$ such that both a and b are in \mathcal{D} and which is also such that, if $\langle a, b \rangle$ is in f , then, for any $c \neq b$, $\langle a, c \rangle$ is not in f .

(Notice that, given these definitions, every function counts as a relation; though not every relation counts as a function.) That is: given any entity in the domain \mathcal{D} , a function associates with that entity *at most one* other thing in \mathcal{D} . It is for this reason that we can write ' $f(a)$ ' and have it mean something exact. If $\langle a, b \rangle$ and $\langle a, c \rangle$ were both in f , then ' $f(a)$ ' would be ambiguous—it might mean b and it might mean c . If $\langle a, b \rangle$ is in the function f , then we may write ' $f(a) = b$ '.

This is how a function works: you hand it some thing, a , and it hands you back some other thing, $f(a)$. Think about the function $f(\mathbf{x}) = \mathbf{x} + 1$, defined on the domain of the natural numbers, $\{1, 2, 3, 4, 5, \dots\}$. If you hand this function 1, then it hands you back $f(1)$, which is just 2. If you hand it 2, then it hands you back $f(2)$, which is just 3. If you hand it 3, then it hands you back $f(3)$, which is just 4. And so on.

A function on the domain \mathcal{D} is *total* iff, no matter what you hand it from the domain \mathcal{D} , it hands you something back.

Total Functions A TOTAL FUNCTION, f , on the domain \mathcal{D} is any function on the domain \mathcal{D} such that, for every a in \mathcal{D} , there is some b in \mathcal{D} such that $\langle a, b \rangle$ is in the function f .

Two sample total functions are shown in figure 6.2. Notice that it does not matter whether everything in the domain may *be handed back*. In figure 6.2b, no matter what you hand f , it hands back b . That doesn't matter. The function is still total, since you can hand it anything in \mathcal{D} , and it will hand you something back.

In this course, we will only be concerned the functions which are *total* on the domain of the interpretation. Thus, an interpretation *does not count* as a *PLI* interpretation unless

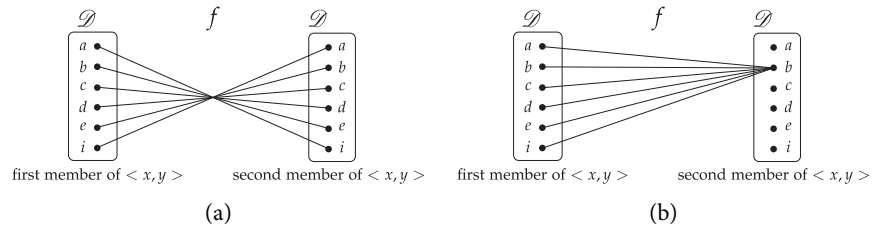


Figure 6.2: Total functions on the domain \mathcal{D}

every function symbol refers to a *total* function on \mathcal{D} . For instance the following is *not* a (partial) *PLI* interpretation:

$$\mathcal{I}^P = \begin{cases} \mathcal{D} = \{1, 2, 3, 4, 5, \dots\} \\ g(\mathbf{x}) = \mathbf{x} - 1 \end{cases} \quad \leftarrow \text{Not a PLI interpretation}$$

This is not a partial *PLI* interpretation because, when you hand g 1, it does not hand you back anything in the domain \mathcal{D} .

In general, we will call the things that you hand a function its *arguments*, and we will call the thing that it hands you back its *value*.

$$f(\underbrace{a}_{\text{argument}}) = \underbrace{b}_{\text{value}}$$

Or, alternatively, since, after all $f(a)$ *just is* (is identical to) b , we may say that $f(a)$ is the value of the argument a .

$$\underbrace{f(\overbrace{a}^{\text{argument}})}_{\text{value}}$$

If f is a 2-place function, then f will have 2 arguments.

$$f(\underbrace{a, b}_{\text{arguments}}) = \underbrace{c}_{\text{value}}$$

Or, alternatively,

$$\underbrace{f(\overbrace{a, b}^{\text{arguments}})}_{\text{value}}$$

If we wish to represent a function of *two* arguments with a set of ordered pairs, then we may do so by utilizing *ordered pairs* of ordered pairs and entities. That is, to represent a function which hands you back c when you hand it the ordered pair $\langle a, b \rangle$, we may

write:

$$f = \{\dots \langle \langle a, b \rangle, c \rangle \dots\}$$

6.1.5 Truth on a *PLI* Interpretation

Suppose that we've got a *PLI* interpretation \mathcal{I} . Then, we can lay down the following rules which tell us what the wffs of *PL* mean on that interpretation—that is, under which conditions they are *true* on that interpretation. (Rules (\mathcal{F}), (\sim), (\vee), ($\&$), (\supset), (\equiv), (\forall), and (\exists) should be familiar from *PL*.)

- (f) If $\ulcorner \mathbf{t}_1 \urcorner, \ulcorner \mathbf{t}_2 \urcorner, \dots, \ulcorner \mathbf{t}_n \urcorner$ are n terms of *PLI* and $\ulcorner \mathbf{f}^n \urcorner$ is an n -place function symbol, then $\ulcorner \mathbf{f}^n(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n) \urcorner$ refers to the value of the things referred to by $\ulcorner \mathbf{t}_1 \urcorner, \ulcorner \mathbf{t}_2 \urcorner, \dots, \ulcorner \mathbf{t}_n \urcorner$, under the function \mathbf{f}^n . (That is: it refers to the thing that \mathbf{f}^n hands you back when you hand it the things denoted by $\ulcorner \mathbf{t}_1 \urcorner, \ulcorner \mathbf{t}_2 \urcorner, \dots, \ulcorner \mathbf{t}_n \urcorner$.)
- (\mathcal{F}) A wff of the form $\ulcorner \mathcal{F}^n \mathbf{t}_1 \dots \mathbf{t}_n \urcorner$ is true on the interpretation \mathcal{I} if the things in the domain denoted by $\ulcorner \mathbf{t}_1 \urcorner \dots \ulcorner \mathbf{t}_n \urcorner$ on the interpretation have the property/bear to each other the relation represented by $\ulcorner \mathcal{F}^n \urcorner$. Otherwise, $\ulcorner \mathcal{F}^n \mathbf{t}_1 \dots \mathbf{t}_n \urcorner$ is false on the interpretation \mathcal{I} .

Note: in both (f) and (\mathcal{F}), ' n ' could be any number greater than or equal to one. If $n = 1$, then the way that I've written the terms, ' $\mathbf{t}_1 \dots \mathbf{t}_n$ ', is misleading, since there is only one term if $n = 1$.

- \Rightarrow) A wff of the form $\ulcorner \mathbf{t}_1 = \mathbf{t}_2 \urcorner$ is true on the interpretation \mathcal{I} if the thing referred to by $\ulcorner \mathbf{t}_1 \urcorner$ is the same thing as the thing referred to by $\ulcorner \mathbf{t}_2 \urcorner$. Otherwise, $\ulcorner \mathbf{t}_1 = \mathbf{t}_2 \urcorner$ is false on the interpretation \mathcal{I} .
- \sim) A wff of the form $\ulcorner \sim \mathbf{P} \urcorner$ is true on the interpretation \mathcal{I} if $\ulcorner \mathbf{P} \urcorner$ is false on the interpretation \mathcal{I} . Otherwise, $\ulcorner \sim \mathbf{P} \urcorner$ is false on the interpretation \mathcal{I} .
- \vee) A wff of the form $\ulcorner \mathbf{P} \vee \mathbf{Q} \urcorner$ is true on the interpretation \mathcal{I} if either $\ulcorner \mathbf{P} \urcorner$ is true on the interpretation \mathcal{I} or $\ulcorner \mathbf{Q} \urcorner$ is true on the interpretation \mathcal{I} . Otherwise, $\ulcorner \mathbf{P} \vee \mathbf{Q} \urcorner$ is false on the interpretation \mathcal{I} .
- $\&$) A wff of the form $\ulcorner \mathbf{P} \& \mathbf{Q} \urcorner$ is true on the interpretation \mathcal{I} if both $\ulcorner \mathbf{P} \urcorner$ is true on the interpretation \mathcal{I} and $\ulcorner \mathbf{Q} \urcorner$ is true on the interpretation \mathcal{I} . Otherwise, $\ulcorner \mathbf{P} \& \mathbf{Q} \urcorner$ is false on the interpretation \mathcal{I} .
- \supset) A wff of the form $\ulcorner \mathbf{P} \supset \mathbf{Q} \urcorner$ is true on the interpretation \mathcal{I} if either $\ulcorner \mathbf{P} \urcorner$ is false on the interpretation \mathcal{I} or $\ulcorner \mathbf{Q} \urcorner$ is true on the interpretation \mathcal{I} . Otherwise, $\ulcorner \mathbf{P} \supset \mathbf{Q} \urcorner$ is false on the interpretation \mathcal{I} .

- \equiv) A wff of the form $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is true on the interpretation \mathcal{I} if both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ have the same truth value on the interpretation \mathcal{I} . Otherwise, $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ is false on the interpretation \mathcal{I} .
- \forall) A wff of the form $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if, for every α in \mathcal{D} , $\lceil \mathbf{P} \rceil$ is true on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$. Otherwise, $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$ is false on the interpretation \mathcal{I} .
- \exists) A wff of the form $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is true on the interpretation \mathcal{I} if for some α in \mathcal{D} , $\lceil \mathbf{P} \rceil$ is true on the \mathbf{x} -variant interpretation $\mathcal{I}_{\mathbf{x} \rightarrow \alpha}$. Otherwise, $\lceil (\exists \mathbf{x})\mathbf{P} \rceil$ is false on the interpretation \mathcal{I} .

For instance, consider the following partial interpretation, \mathcal{I}^P :

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \{1, 2\} \\ f & = \{ \langle 1, 2 \rangle, \langle 2, 1 \rangle \} \end{cases}$$

Let's check whether the wff

$$(\forall x)x = f(f(x))$$

is true on this interpretation. This wff is of the form $\lceil (\forall \mathbf{x})\mathbf{P} \rceil$; that is, it is a universally quantified wff—the wff's main operator is $\lceil (\forall \mathbf{x}) \rceil$. The semantic rule (\forall) tells us that this wff is true iff $\lceil x = f(f(x)) \rceil$ is true on every x -variant of \mathcal{I}^P . There are two x -variants of \mathcal{I}^P :

$$\mathcal{I}_{x \rightarrow 1}^P \quad \text{and} \quad \mathcal{I}_{x \rightarrow 2}^P$$

Start with the first one, $\mathcal{I}_{x \rightarrow 1}^P$. Is $\lceil x = f(f(x)) \rceil$ true on this interpretation? It is true on this interpretation iff the thing denoted by $\lceil x \rceil$ on this interpretation is identical to the thing denoted by $\lceil f(f(x)) \rceil$ on this interpretation. x denotes the number 1. So we just have to check to see whether $\lceil f(f(x)) \rceil$ denotes 1 in order to see whether this wff is true or false. The semantic rule (**f**) tells us that $\lceil f(f(x)) \rceil$ refers to the value of whatever $f(x)$ refers to, under the function f . But we don't yet know what $\lceil f(x) \rceil$ refers to on $\mathcal{I}_{x \rightarrow 1}^P$. Well, the semantic rule (**f**) tells us that, on the interpretation $\mathcal{I}_{x \rightarrow 1}^P$, $\lceil f(x) \rceil$ refers to the value of whatever x refers to under the function f . We know that x refers to 1 on $\mathcal{I}_{x \rightarrow 1}^P$. And we know that $f(1) = 2$ on $\mathcal{I}_{x \rightarrow 1}^P$. So $\lceil f(x) \rceil$ refers to 2 on $\mathcal{I}_{x \rightarrow 1}^P$. And we know that $f(2) = 1$ on $\mathcal{I}_{x \rightarrow 1}^P$, so we know that $\lceil f(f(x)) \rceil$ refers to 1 on $\mathcal{I}_{x \rightarrow 1}^P$. And this is identical to the thing that x refers to on $\mathcal{I}_{x \rightarrow 1}^P$. So $\lceil x = f(f(x)) \rceil$ is true on $\mathcal{I}_{x \rightarrow 1}^P$.

Consider next $\mathcal{I}_{x \rightarrow 2}^P$. Is $\lceil x = f(f(x)) \rceil$ true on this interpretation? It is true on this interpretation iff the thing denoted by $\lceil x \rceil$ on this interpretation is identical to the thing denoted by $\lceil f(f(x)) \rceil$ on this interpretation. $\lceil f(x) \rceil$ refers to the value of 2 under the function f , which is 1. So $\lceil f(x) \rceil$ refers to 1. And, therefore, $\lceil f(f(x)) \rceil$ refers to the value of 1 under the function f , which is 2. So $\lceil f(f(x)) \rceil$ refers to 2. So $\lceil x = f(f(x)) \rceil$ is true on the interpretation $\mathcal{I}_{x \rightarrow 2}^P$.

So ' $x = f(f(x))$ ' is true on both $\mathcal{I}_{x \rightarrow 1}^P$ and $\mathcal{I}_{x \rightarrow 2}^P$. But these are all of the x -variants of \mathcal{I}^P . So ' $(\forall x)x = f(f(x))$ ' is true on the interpretation \mathcal{I}^P .

6.2 Logical Notions of *PLI*

All of the logical notions of *PLI* will be exactly the same as they were in *PL*, once we swap out the notion of a *PL* interpretation for that of a *PLI* interpretation. Thus,

PLI Validity A *PLI* argument is *PLI* VALID iff there is no *PLI* interpretation on which the premises are all true while the conclusion is false.

PLI INVALIDITY A *PLI* argument is *PLI* INVALID iff there is some *PLI* interpretation on which the premises are all true while the conclusion is false.

PLI CONSISTENCY A set of wffs of *PLI* $\{\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner\}$ is *PLI* CONSISTENT iff there is some *PLI* interpretation on which $\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner$ are all true.

PLI INCONSISTENCY A set of wffs of *PLI* $\{\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner\}$ is *PLI* INCONSISTENT iff there is no *PLI* interpretation on which $\ulcorner \mathbf{A}_1 \urcorner, \ulcorner \mathbf{A}_2 \urcorner, \dots, \ulcorner \mathbf{A}_N \urcorner$ are all true.

PLI EQUIVALENCE A pair of wffs of *PLI*, $\ulcorner \mathbf{A} \urcorner$ and $\ulcorner \mathbf{B} \urcorner$, are *PLI* EQUIVALENT iff $\ulcorner \mathbf{A} \urcorner$ and $\ulcorner \mathbf{B} \urcorner$ are true in all the same *PLI* interpretations and false in all the same *PLI* interpretations (*i.e.*, iff there is no *PLI* interpretation on which $\ulcorner \mathbf{A} \urcorner$ and $\ulcorner \mathbf{B} \urcorner$ have different truth-values).

PLI Tautology A wff of *PLI* $\ulcorner \mathbf{A} \urcorner$ is a *PLI* TAUTOLOGY iff there is no *PLI* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is false (*i.e.*, iff $\ulcorner \mathbf{A} \urcorner$ is true on every *PLI* interpretation).

PLI CONTRADICTION A wff of *PLI* $\ulcorner \mathbf{A} \urcorner$ is a *PLI* CONTRADICTION iff there is no *PLI* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is true (*i.e.*, iff $\ulcorner \mathbf{A} \urcorner$ is false on every *PLI* interpretation).

PLI CONTINGENCY A wff of *PLI* $\ulcorner \mathbf{A} \urcorner$ is a *PLI* CONTINGENCY iff there is some *PLI* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is true and some *PLI* interpretation on which $\ulcorner \mathbf{A} \urcorner$ is false.

6.3 Trees for *PLI*

We may test for all of these logical properties of *PLI* just as we did with *PL*: with the aid of trees. However, in *PLI*, we will need to add two additional rules and modify our account of what it is for a tree to be *complete*.

6.3.1 The Rule (=)

Our first new rule tells us that, when we know that the thing denoted by $\ulcorner t_1 \urcorner$ is identical to the thing denoted by $\ulcorner t_2 \urcorner$, we may substitute $\ulcorner t_2 \urcorner$ for $\ulcorner t_1 \urcorner$ wherever $\ulcorner t_1 \urcorner$ occurs (and *vice versa*).

$\underline{=}$ $\mathbf{P}[t_1]$ $t_1 = t_2$ $ $ $\mathbf{P}[t_1 \rightarrow t_2]$ <p style="text-align: center;">and</p> $\mathbf{P}[t_2]$ $t_1 = t_2$ $ $ $\mathbf{P}[t_2 \rightarrow t_1]$
--

Here's how to read this rule: if you have a wff $\ulcorner \mathbf{P}[t_1] \urcorner$ in which the constant or free variable $\ulcorner t_1 \urcorner$ occurs on an open branch of the tree, and you have a wff of the form $\ulcorner t_1 = t_2 \urcorner$ on the same branch, then you may substitute $\ulcorner t_2 \urcorner$ for $\ulcorner t_1 \urcorner$ in $\ulcorner \mathbf{P}[t_1] \urcorner$, writing down the wff $\ulcorner \mathbf{P}[t_1 \rightarrow t_2] \urcorner$. You may similarly substitute $\ulcorner t_1 \urcorner$ for $\ulcorner t_2 \urcorner$ in any wff in which $\ulcorner t_2 \urcorner$ appears. That is: it does not matter whether the terms are on the left- or right-hand-side of the equals sign.

Note: when you apply this rule, you do not check off the wff $\ulcorner t_1 = t_2 \urcorner$, and you do not check off any other wff either. You may have to make multiple substitutions before you are done with the tree. In fact, an open tree will not be complete until you have made *all the possible* substitutions which are allowed by the rule (=).

For instance, suppose that we wish to check the following *PLI* argument for *PTI*-validity:

$$(\forall x)(\forall y)x = y / (\exists w)Pw // (\forall z)Pz$$

To do so, we start a tree with the wffs $\ulcorner (\forall x)(\forall y)x = y \urcorner$, $\ulcorner (\exists w)Pw \urcorner$ and $\ulcorner \sim(\forall z)Pz \urcorner$:

$$\begin{array}{c} (\forall x)(\forall y)x = y \\ (\exists w)Pw \\ \sim(\forall z)Pz \end{array}$$

We may begin by applying the rule for $(\sim\forall)$ to $\ulcorner \sim(\forall z)Pz \urcorner$, as shown:

$$\begin{array}{c}
 (\forall x)(\forall y)x = y \\
 (\exists w)Pw \\
 \sim(\forall z)Pz \quad \checkmark \\
 | \\
 (\exists z)\sim Pz
 \end{array}$$

And we may then apply the rule (\exists) to $(\exists w)Pw$ and $(\exists z)\sim Pz$:

$$\begin{array}{c}
 (\forall x)(\forall y)x = y \\
 (\exists w)Pw \quad \checkmark \\
 \sim(\forall z)Pz \quad \checkmark \\
 | \\
 (\exists z)\sim Pz \quad \checkmark \\
 | \\
 Pa \\
 | \\
 \sim Pb
 \end{array}$$

When we then apply the rule (\forall) to $(\forall x)(\forall y)x = y$ by writing down the substitution instance $(\forall y)a = y$ and apply the rule (\forall) to $(\forall y)a = y$ by writing down the substitution instance $a = b$, we get:

$$\begin{array}{c}
 (\forall x)(\forall y)x = y \quad \textcircled{a} \\
 (\exists w)Pw \quad \checkmark \\
 \sim(\forall z)Pz \quad \checkmark \\
 | \\
 (\exists z)\sim Pz \quad \checkmark \\
 | \\
 Pa \\
 | \\
 \sim Pb \\
 | \\
 (\forall y)a = y \quad \textcircled{b} \\
 | \\
 a = b
 \end{array}$$

Now, we may use the rule $(=)$ to substitute $'a'$ for $'b'$ in $\sim Pb$, writing down $\sim Pa$:

$$\begin{array}{c}
 (\forall x)(\forall y)x = y \text{ (a)} \\
 (\exists w)Pw \checkmark \\
 \sim(\forall z)Pz \checkmark \\
 | \\
 (\exists z)\sim Pz \checkmark \\
 | \\
 Pa \\
 | \\
 \sim Pb \\
 | \\
 (\forall y)a = y \text{ (b)} \\
 | \\
 a = b \\
 | \\
 \sim Pa \\
 \times
 \end{array}$$

Since ' Pa ' and ' $\sim Pa$ ' appear on the same branch, this branch closes, and the tree closes. So the *PLI* argument

$$(\forall x)(\forall y)x = y / (\exists w)Pw // (\forall z)Pz$$

is *PTI*-valid.

6.3.2 The Rule ($\neq \times$)

Our next rule tells us that, if, on any branch, a wff of the form ' $\mathbf{t} \neq \mathbf{t}$ ' appears, for any term ' \mathbf{t} ', then you may immediately close that branch of the tree.

$$\begin{array}{c}
 \hline
 (\neq \times) \\
 \hline
 \mathbf{t} \neq \mathbf{t} \\
 | \\
 \times
 \end{array}$$

For instance, suppose that we wish to see whether the following wff is a *PTI*-contradiction:

$$(\exists x)(\forall y)f(x) \neq y$$

To check this, we begin a tree with ' $(\exists x)(\forall y)f(x) \neq y$ ' at its root. We may begin by applying the rule (\exists), writing down a substitution instance of this wff which utilizes an entirely new constant, like so:

$$\begin{array}{c} (\exists x)(\forall y)f(x) \neq y \quad \checkmark \\ | \\ (\forall y)f(a) \neq y \end{array}$$

Now, we may utilize the rule (\forall) and substitute the term ' $f(a)$ ' for ' y ' in ' $(\forall y)f(a) \neq y$ ':

$$\begin{array}{c} (\exists x)(\forall y)f(x) \neq y \quad \checkmark \\ | \\ (\forall y)f(a) \neq y \quad \textcircled{f(a)} \\ | \\ f(a) \neq f(a) \\ \times \end{array}$$

Since ' $f(a) \neq f(a)$ ' appears on the only open branch of the tree, we may close that branch by the rule ($\neq \times$). Thus, the tree closes, and

$$(\exists x)(\forall y)f(x) \neq y$$

is a *PTI*-contradiction. (It is a *PLI* contradiction, too, because we require that every function f be *total* on the domain of the interpretation—therefore, for every thing in the domain, x , $f(x)$ must be defined and must be within the domain. So there is no thing in the domain x such that $f(x)$ is distinct from every thing in the domain.)

6.3.3 Completing Trees

You may have been surprised, in the previous tree, to see me instantiate the term ' $f(a)$ '. This was allowed by the rule (\forall), which allows us to write down a substitution instance $\mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}]$ of a universally quantified wff ' $(\forall \mathbf{x})\mathbf{P}$ ', for any term ' \mathbf{t} '.

$$\begin{array}{c} \underline{(\forall)} \\ (\forall \mathbf{x})\mathbf{P} \quad \textcircled{\mathbf{t}} \\ | \\ \mathbf{P}[\mathbf{x} \rightarrow \mathbf{t}] \end{array}$$

for any term \mathbf{t}

Since ' $f(a)$ ' is a term, we may write down a substitution instance of ' $(\forall y)f(a) \neq y$ ' in which we replace the ' y ' bound by ' $(\forall y)$ ' with ' $f(a)$ '.

However, we must now alter our account of what it is for a tree to be *complete*. Our old account, recall, went like this:

To complete a tree:

1. Apply the relevant rules to all wffs appearing on open branches, in any order you like.
2. If a wff $\lceil \mathbf{P} \rceil$ and its negation $\lceil \sim \mathbf{P} \rceil$ appear on the same branch, then close that branch by writing ‘ \times ’ at the bottom of the branch.
3. If every branch closes, then you are done; in this case, we say that *the tree closes*.
4. If you have applied every relevant rule to every wff on every open branch which is not atomic or the negation of an atomic wff, **and, you have applied the rule (\forall) to every universally quantified wff appearing on an open branch by instantiating every constant or free variable which appears on an open branch with that universally quantified wff**, then you are done; if, after doing this, there remains an open branch, then we say that *the tree remains open*.

In this account of what it is to complete a tree, we said only that you must write down a substitution instance of a universally quantified wff *for every constant or free variable*. This leaves out the substitution instances of terms like ‘ $f(a)$ ’. So, if we kept this account of what it is for a tree to be complete, we would have to count the following tree as complete:

$$\begin{array}{c}
 (\exists x)(\forall y)f(x) \neq y \checkmark \\
 | \\
 (\forall y)f(a) \neq y \textcircled{a} \\
 | \\
 f(a) \neq a
 \end{array}
 \quad \leftarrow \quad \text{This tree is not complete!}$$

But this would tell us that ‘ $(\exists x)(\forall y)f(x) \neq y$ ’ is not a *PTI*-contradiction. However, *it is a PLI-contradiction*. The reason is that we required, in our definition of a *PLI* interpretation, that every function be *total*. But, if every function is total, then there is no thing ‘ x ’ could refer to in the domain that would make ‘ $(\forall y)f(x) \neq y$ ’. For if the function f is total, then ‘ $f(x)$ ’ must refer to something in the domain \mathcal{D} . But if ‘ y ’ refers to that thing, then ‘ $f(x) \neq y$ ’ would be false. So, no matter what ‘ x ’ refers to, there’s something y could refer to which would make ‘ $f(x) \neq y$ ’ false. So the universally quantified wff ‘ $(\forall y)f(x) \neq y$ ’ is false on every x -variant interpretation; so the existentially quantified wff ‘ $(\exists x)(\forall y)f(x) \neq y$ ’ must be false.

We will fix this by introducing a new concept: that of a *constant term*. A term in a wff is a *constant term* iff it is a term that does not contain any bound variables.

A term of *PLI*, $\ulcorner t \urcorner$, in a wff of *PLI*, is a CONSTANT TERM if and only if no bound variables occur within $\ulcorner t \urcorner$

For instance, in the wff

$$(\forall x)Rf(x, a)g(y)$$

' $f(x, a)$ ' is *not* a constant term, since the bound variable ' x ' occurs in that term. On the other hand, ' $g(y)$ ' is a constant term, since only the free variable ' y ' occurs in that term. Similarly, in the wff:

$$(\exists x)Gh(x)f(a, b)$$

' $h(x)$ ' is *not* a constant term, since the bound variable ' x ' appears within it. ' $f(a, b)$ ', on the other hand *is* a constant term.

With this definition, we may emend our account of what it is for a tree to be complete as follows.

To complete a tree:

1. Apply the relevant rules to all wffs appearing on open branches, in any order you like.
2. If a wff $\ulcorner P \urcorner$ and its negation $\ulcorner \sim P \urcorner$ appear on the same branch, then close that branch by writing ' \times ' at the bottom of the branch.
3. If a wff of the form $\ulcorner t \neq t \urcorner$ appears at any point on a branch, then close that branch by writing ' \times ' at the bottom of the branch.
4. If every branch closes, then you are done; in this case, we say that *the tree closes*.
5. If:
 - (a) you have applied every relevant rule to every wff on every open branch which is not atomic or the negation of an atomic wff;
 - (b) you have made all of the substitutions licensed by the rule (=); and
 - (c) you have applied the rule (\forall) to every universally quantified wff appearing on an open branch by instantiating every constant term or free variable which appears on an open branch with that universally quantified wff;

then you are done; if, after doing this, there remains an open branch, then we say that *the tree remains open*.

For instance, consider the following tree:

$$\begin{array}{c}
 (\exists x)f(x) = x \checkmark \\
 (\forall x)x = x \textcircled{a} \\
 | \\
 f(a) = a \\
 | \\
 a = a
 \end{array}$$

This tree is not complete because, though we have substituted the constant ‘ a ’ for ‘ x ’ in ‘ $(\forall x)x = x$ ’, this is not the only constant term appearing on the same branch as ‘ $(\forall x)x = x$ ’. There is also the constant term ‘ $f(a)$ ’. This must be substituted too before the tree is complete.

$$\begin{array}{c}
 (\exists x)f(x) = x \checkmark \\
 (\forall x)x = x \textcircled{a} \textcircled{f(a)} \\
 | \\
 f(a) = a \\
 | \\
 a = a \\
 | \\
 f(a) = f(a)
 \end{array}$$

We must also make sure that we have made all the substitutions licensed by the rule (=) before the tree is complete. ‘ $a = a$ ’ tells us that we may substitute ‘ a ’ for ‘ a ’ anywhere it appears. However, doing this would just give us back a wff which already appears on the branch. *In general, you do not have to make the substitutions licensed by (=) which result in wffs already on the branch. However, if a substitution does not already appear on the branch, then you must make it before the tree is complete.* That means, in particular, that on the tree above, we must substitute ‘ a ’ for the first ‘ $f(a)$ ’ in ‘ $f(a) = f(a)$ ’, though we needn’t substitute it for the first. When we make the requisite substitution, we get the following tree, which is complete:

$$\begin{array}{c}
 (\exists x)f(x) = x \checkmark \\
 (\forall x)x = x \textcircled{a} \textcircled{f(a)} \\
 | \\
 f(a) = a \\
 | \\
 a = a \\
 | \\
 f(a) = f(a) \\
 | \\
 a = f(a) \\
 \circ
 \end{array}$$

6.4 PTI Consistency

As before, if a tree beginning with all and only the wffs in a set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N\}$ closes, then the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N\}$ is *PTI*-inconsistent. If a completed tree beginning with all and only the wffs in the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N\}$ remains open, then the set $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_N\}$ is *PTI*-consistent.

Remember: it is important, in the definition of a *PTI*-consistent set of wffs, that the open tree be *complete*. That means you must have checked off every wff which may be checked off, you must make all of the relevant substitutions licensed by the rule (=), and you must substitute all of the constant terms appearing on the same branch as a universally quantified wff.

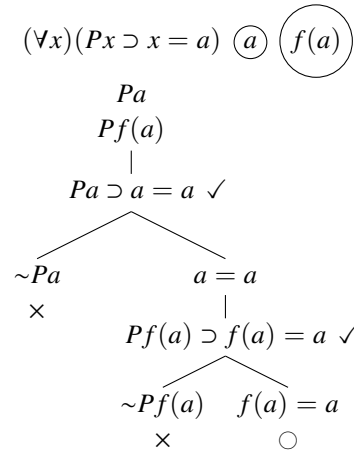
Is the set $\{(\forall x)(Px \supset x = a), Pa, Pf(a)\}$ *PTI*-consistent? We begin the tree with $(\forall x)(Px \supset x = a)$, Pa , and $Pf(a)$ at its root:

$$\begin{array}{c} (\forall x)(Px \supset x = a) \\ Pa \\ Pf(a) \end{array}$$

We must apply a rule to $(\forall x)(Px \supset x = a)$ by instantiating some constant term for ' x '. ' a ' appears on the tree with $(\forall x)(Px \supset x = a)$, so it makes sense to instantiate ' a ':

$$\begin{array}{c} (\forall x)(Px \supset x = a) \text{ (a)} \\ Pa \\ Pf(a) \\ | \\ Pa \supset a = a \checkmark \\ \wedge \\ \sim Pa \quad a = a \\ \times \end{array}$$

The tree has not closed, but this does not tell us that $\{(\forall x)(Px \supset x = a), Pa, Pf(a)\}$ is *PTI*-consistent, since the tree is not yet complete. The constant term ' $f(a)$ ' also appears on the same branch as $(\forall x)(Px \supset x = a)$, so we must instantiate it as well. When we do so, we get



The tree is now complete, and it remains open. So we can conclude that $\{(\forall x)(Px \supset x = a), Pa, Pf(a)\}$ is *PTI*-consistent.

6.5 *PTI* Validity

As before, if a tree with the premises of an argument and the negation of the conclusion of an argument at its root closes, then the argument is *PTI*-valid. If a completed tree with the premises of the argument and the negation of the conclusion at its root remains open, then the argument is *PTI*-invalid.

For instance, consider the *PLI* argument

$$(\forall x)(\forall y)x = y \ // \ (\forall z)(\forall w)(Rzw \supset R wz)$$

To see whether this argument is *PTI* valid, we begin a tree with the premise, ' $(\forall x)(\forall y)x = y$ ' and the negation of its conclusion, ' $\sim(\forall z)(\forall w)(Rzw \supset R wz)$ ' at its root, as shown:

$$\begin{array}{c}
 (\forall x)(\forall y)x = y \\
 \sim(\forall z)(\forall w)(Rzw \supset R wz)
 \end{array}$$

We may begin by applying the rule ($\sim\forall$) to ' $\sim(\forall z)(\forall w)(Rzw \supset R wz)$ ':

$$\begin{array}{c}
(\forall x)(\forall y)x = y \\
\sim(\forall z)(\forall w)(Rzw \supset R wz) \checkmark \\
| \\
(\exists z)\sim(\forall w)(Rzw \supset R wz) \checkmark \\
| \\
\sim(\forall w)(Raw \supset Rwa) \checkmark \\
| \\
(\exists w)\sim(Raw \supset Rwa) \checkmark \\
| \\
\sim(Rab \supset Rba) \\
| \\
Rab \\
\sim Rba
\end{array}$$

We may now apply the rule (\forall) to ' $(\forall x)(\forall y)x = y$ ' by writing down ' $(\forall y)a = y$ '; and then applying the rule (\forall) to ' $(\forall y)a = y$ ' by writing down ' $a = b$ '.

$$\begin{array}{c}
(\forall x)(\forall y)x = y \text{ (a)} \\
\sim(\forall z)(\forall w)(Rzw \supset R wz) \checkmark \\
| \\
(\exists z)\sim(\forall w)(Rzw \supset R wz) \checkmark \\
| \\
\sim(\forall w)(Raw \supset Rwa) \checkmark \\
| \\
(\exists w)\sim(Raw \supset Rwa) \checkmark \\
| \\
\sim(Rab \supset Rba) \\
| \\
Rab \\
\sim Rba \\
| \\
(\forall y)a = y \text{ (b)} \\
| \\
a = b
\end{array}$$

At this point, we may substitute ' a ' for ' b ' in ' Rab ', and substitute ' b ' for ' a ' in ' Rba ', at which point the tree closes.

$$\begin{array}{c}
(\forall x)(\forall y)x = y \text{ (a)} \\
\sim(\forall z)(\forall w)(Rzw \supset R wz) \checkmark \\
| \\
(\exists z)\sim(\forall w)(Rzw \supset R wz) \checkmark \\
| \\
\sim(\forall w)(Raw \supset Rwa) \checkmark \\
| \\
(\exists w)\sim(Raw \supset Rwa) \checkmark \\
| \\
\sim(Rab \supset Rba) \\
| \\
Rab \\
\sim Rba \\
| \\
(\forall y)a = y \text{ (b)} \\
| \\
a = b \\
| \\
Raa \\
| \\
\sim Raa \\
\times
\end{array}$$

Thus, the *PLI* argument

$$(\forall x)(\forall y)x = y // (\forall z)(\forall w)(Rzw \supset R wz)$$

is *PTI*-valid.

6.6 *PTI* Tautologies, Contingencies, and Contradictions

As before, if a tree with $\lceil \sim \mathbf{P} \rceil$ at its root closes, then $\lceil \mathbf{P} \rceil$ is a *PTI*-tautology. For instance, suppose that we start a tree with $\lceil \sim(\exists x)x = x \rceil$ at its root:

$$\begin{array}{c}
\sim(\exists x)x = x \checkmark \\
| \\
(\forall x)x \neq x \text{ (a)} \\
| \\
a \neq a \\
\times
\end{array}$$

Because the tree closes, this shows us that $\lceil (\exists x)x = x \rceil$ is a *PLI*-tautology. The reason that this is a tautology is that we do not allow our domains to be empty. So there must always be something in the domain, and that thing will always be identical to itself.

6.6.1 Properties of Identity

By utilizing *PLI* trees, we may also establish some tautologies which tell us some interesting properties of the identity relation, =. Identity is what's known as an *equivalence* relation. An equivalence relation is any relation which satisfies the following three properties.

Reflexivity R is a reflexive relation on the domain \mathcal{D} iff, for all x in \mathcal{D} , Rxx .

$$(\forall x)Rxx$$

Symmetry R is a symmetric relation on the domain \mathcal{D} iff, for all x and y in \mathcal{D} , if Rxy , then Ryx .

$$(\forall x)(\forall y)(Rxy \supset Ryx)$$

Transitivity R is a transitive relation on the domain \mathcal{D} iff, for all x , y , and z in \mathcal{D} , if Rxy and Ryz , then Rxz .

$$(\forall x)(\forall y)(\forall z)((Rxy \& Ryz) \supset Rxz)$$

We may show that = has each of these three properties with *PLI* trees. We can show that identity is reflexive by showing that ' $(\forall x)x = x$ ' is a *PTI*-tautology. When we start a tree with its negation, we get the following.

$$\begin{array}{c} \sim(\forall x)x = x \checkmark \\ | \\ (\exists x)x \neq x \checkmark \\ | \\ a \neq a \\ \times \end{array}$$

Similarly, we can show that identity is symmetric by showing that ' $(\forall x)(\forall y)(x = y \supset y = x)$ ' is a *PTI*-tautology with the following tree.

$$\begin{array}{c}
\sim(\forall x)(\forall y)(x = y \supset y = x) \checkmark \\
| \\
(\exists x)\sim(\forall y)(x = y \supset y = x) \checkmark \\
| \\
\sim(\forall y)(a = y \supset y = a) \checkmark \\
| \\
(\exists y)\sim(a = y \supset y = a) \checkmark \\
| \\
\sim(a = b \supset b = a) \checkmark \\
| \\
a = b \\
b \neq a \\
| \\
a \neq a \\
\times
\end{array}$$

Finally, we can show that = is transitive by showing that ' $(\forall x)(\forall y)(\forall z)((x = y \& y = z) \supset x = z)$ ' is a *PLI*-tautology with the following tree:

$$\begin{array}{c}
\sim(\forall x)(\forall y)(\forall z)((x = y \& y = z) \supset x = z) \checkmark \\
| \\
(\exists x)\sim(\forall y)(\forall z)((x = y \& y = z) \supset x = z) \checkmark \\
| \\
\sim(\forall y)(\forall z)((a = y \& y = z) \supset a = z) \checkmark \\
| \\
(\exists y)\sim(\forall z)((a = y \& y = z) \supset a = z) \checkmark \\
| \\
\sim(\forall z)((a = b \& b = z) \supset a = z) \checkmark \\
| \\
(\exists z)\sim((a = b \& b = z) \supset a = z) \checkmark \\
| \\
\sim((a = b \& b = c) \supset a = c) \checkmark \\
| \\
a = b \& b = c \checkmark \\
a \neq c \\
| \\
a = b \\
b = c \\
| \\
a = c \\
\times
\end{array}$$

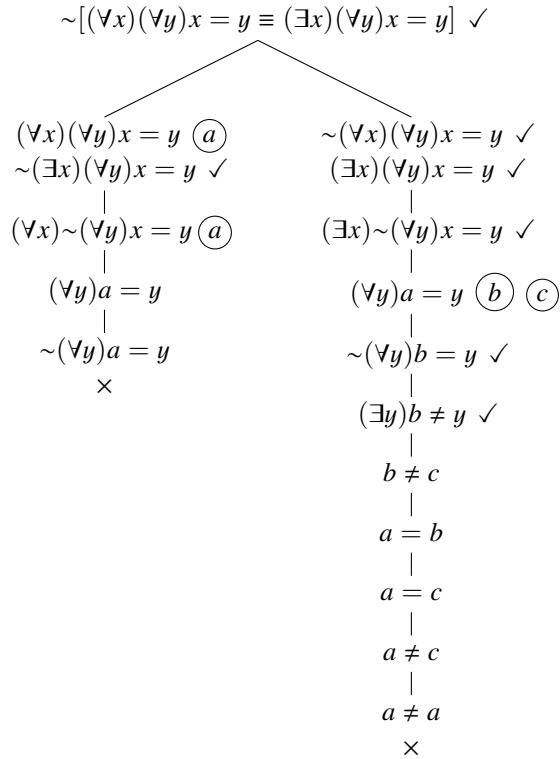
6.7 PTI Equivalence

If a tree starting with the sole wff $\lceil \sim(\mathbf{P} \equiv \mathbf{Q}) \rceil$ at its root closes, then $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are *PTI*-equivalent. If a tree starting with the sole wff $\lceil \sim(\mathbf{P} \equiv \mathbf{Q}) \rceil$ at its root remains open, then $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ are not *PTI*-equivalent.

For instance, consider the wffs

$$(\forall x)(\forall y)x = y \quad \text{and} \quad (\exists x)(\forall y)x = y$$

The first wff says “Everything in the domain is identical to everything in the domain.” The second wff says “There’s something in the domain which is identical to everything in the domain.” The first wff could only be true if there is exactly one thing in the domain; and the second wff similarly could only be true if there’s exactly one thing in the domain. So we should expect these two wffs to be *PTI*-equivalent (if the tree method is any good). And, indeed, they are. Here is the completed tree with $\lceil \sim[(\forall x)(\forall y)x = y \equiv (\exists x)(\forall y)x = y] \rceil$ at its root:



6.8 Infinite *PLI* Trees

Consider the following *PLI* tree which begins with the wff ‘ $(\exists x)(\forall y)f(y) \neq x$ ’ at its root.

$$\begin{array}{c}
 (\exists x)(\forall y)f(y) \neq x \checkmark \\
 | \\
 (\forall y)f(y) \neq a \textcircled{a} \\
 | \\
 f(a) \neq a
 \end{array}$$

Is this tree complete? It is not. Though the constant ‘ a ’ has been instantiated for ‘ y ’ in ‘ $(\forall y)f(y) \neq a$ ’, the constant term ‘ $f(a)$ ’—which also appears on a branch with ‘ $(\forall y)f(y) \neq a$ ’—has not. In order for the tree to be complete, we must instantiate this constant term for ‘ y ’ in ‘ $(\forall y)f(y) \neq a$ ’ as well.

$$\begin{array}{c}
 (\exists x)(\forall y)f(y) \neq x \checkmark \\
 | \\
 (\forall y)f(y) \neq a \textcircled{a} \textcircled{f(a)} \\
 | \\
 f(a) \neq a \\
 | \\
 f(f(a)) \neq a
 \end{array}$$

Is the tree complete now? No. The new constant term ‘ $f(f(a))$ ’ now appears on the same branch as ‘ $(\forall y)f(y) \neq a$ ’. So this term, too, must be instantiated for ‘ y ’ in ‘ $(\forall y)f(y) \neq a$ ’. And when we do this, we will have a new constant term, ‘ $f(f(f(a)))$ ’. And it will go on like this, forever. So the completed tree will end up being infinitely long.

$$\begin{array}{c}
 (\exists x)(\forall y)f(y) \neq x \checkmark \\
 | \\
 (\forall y)f(y) \neq a \textcircled{a} \textcircled{f(a)} \textcircled{f(f(a))} \textcircled{f(f(f(a)))} \dots \\
 | \\
 f(a) \neq a \\
 | \\
 f(f(a)) \neq a \\
 | \\
 f(f(f(a))) \neq a \\
 | \\
 f(f(f(f(a)))) \neq a \\
 \vdots
 \end{array}$$

This tree is complete. For every constant term which appears on the branch will be instantiated for 'y' in ' $(\forall y)f(y) \neq a$ ' at some point on the tree. Since the tree does not close, we can conclude that ' $(\exists x)(\forall y)f(y) \neq x$ ' is not a *PTI*-contradiction.

6.9 Number Claims

At Least Two

Suppose that we wish to say something that is true in all and only the interpretations in which there are two distinct things in the domain of the interpretation. We may do this as follows:

$$(\exists x)(\exists y)x \neq y$$

This is true on an interpretation \mathcal{I} if and only if $(\exists y)x \neq y$ is true on some x -variant interpretation $\mathcal{I}_{x \rightarrow \alpha}$. And $(\exists y)x \neq y$ is true on the x -variant interpretation $\mathcal{I}_{x \rightarrow \alpha}$ if and only if $x \neq y$ is true on some y -variant of that interpretation, $\mathcal{I}_{x \rightarrow \alpha, y \rightarrow \beta}$. So $(\exists x)(\exists y)x \neq y$ is true if and only if there are two distinct things in our domain such that x can refer to one of them and y can refer to the other. So $(\exists x)(\exists y)x \neq y$ is true if and only if there are at least two things in the model. $(\exists x)(\exists y)x \neq y$ is *not* a *PLI*-tautology, since we don't require that our domains have more than one thing in them.

Here, we don't need to additionally specify that $y \neq x$, since this follows from the symmetry of \neq . From ' $(\exists x)(\exists y)x \neq y$ ', it follows that ' $(\exists x)(\exists y)(x \neq y \ \& \ y \neq x)$ ', as the following tree demonstrates:

$$\begin{array}{c}
(\exists x)(\exists y)x \neq y \checkmark \\
\sim(\exists x)(\exists y)(x \neq y \& y \neq x) \checkmark \\
| \\
(\exists y)a \neq y \checkmark \\
| \\
a \neq b \\
| \\
(\forall x)\sim(\exists y)(x \neq y \& y \neq x) \textcircled{a} \textcircled{b} \\
| \\
\sim(\exists y)(a \neq y \& y \neq a) \checkmark \\
| \\
(\forall y)\sim(a \neq y \& y \neq a) \textcircled{b} \\
| \\
\sim(a \neq b \& b \neq a) \checkmark \\
\swarrow \quad \searrow \\
\sim a \neq b \quad \sim b \neq a \checkmark \\
\times \quad | \\
\quad b = a \\
\quad | \\
\quad a \neq a \\
\quad \times
\end{array}$$

At Least Three

Similarly, if we wish to say something that is true in all and only the *PLI*-interpretations in which there are three distinct things in the domain, we may say the following:

$$(\exists x)(\exists y)(\exists z)((x \neq y \& y \neq z) \& x \neq z)$$

Here, it is not enough to say merely that $x \neq y$ and that $y \neq z$. For it could still be that $x = z$. So we must rule this out by specifying that $x \neq z$. The wff above says that the domain contains three things, all of which are distinct from one another. Of course, there could be other things besides these three. So what the wff above says is just that there are *at least* three things in the domain.

At Least Four

We could go on. Suppose that we wish to say something that's true in all and only the *PLI*-interpretations on which there are at least *four* things in the domain. Then, we could say the following:

$$(\exists x)(\exists y)(\exists z)(\exists w)((((x \neq y \& x \neq z) \& x \neq w) \& y \neq z) \& y \neq w) \& z \neq w)$$

A CHALLENGE

See if you can come up with a wff of *PLI* that's true if and only if there are *infinitely many* things in the domain. (You'll have to use more than identity. *Hint*: to think it through, use an interpretation whose domain is the counting numbers, use the predicate Gxy for 'x is greater than y ($x > y$)', and try to think of a collection of claims which will tell you that there are an infinite number of numbers. *A further hint*: translate the claims "nothing is greater than itself", "every number has some number that's greater than it", and then think about how, with the foregoing claims laid down, you could rule out 'loops' of greatness, like, *e.g.*, Gab , Gbc , and Gca .)

At Least Two P s

Now, suppose that we wish to say, not just that there's at least two *things* in the domain, but additionally, that there's at least two things *that are P* in the domain. We can say that by saying, first, that there's at least two things, and next, that those two things are P :

$$(\exists x)(\exists y)((x \neq y \ \& \ Px) \ \& \ Py)$$

At Least Three P s

Similarly, suppose that we wish to say that there are at least three things *that are P* in the domain. We can say that by saying, first, that there's at least three things, and next, that those three things are P :

$$(\exists x)(\exists y)(\exists z)((((x \neq y \ \& \ y \neq z) \ \& \ x \neq z) \ \& \ Px) \ \& \ Py) \ \& \ Pz)$$

No More Than One

The previous translations put a *lower* bound on the number of things in the domain. Suppose that, instead, we wish to be an *upper* bound on the number of things in the domain. Suppose that we wish to say that there is *no more than one* thing in the domain. If we wish to say this, then we could just say that there's *something* which everything in the domain is identical to. If everything in the domain is the same as that one thing, then there can only be one thing in the domain. Thus,

$$(\exists x)(\forall y)y = x$$

translates “there is at least one thing in the domain.” (Note: since we require the domain to contain at least one thing, this is equivalent to saying that there is *exactly* one thing in the domain.)

No More Than Two

Suppose, on the other hand, that we wish to say that there are no more than two things in the domain. We may say that by saying that there are two things x and y such that, for any thing in the domain z , z is either identical to x or z is identical to y .

$$(\exists x)(\exists y)(\forall z)(z = x \vee z = y)$$

This could also be true if there is but one thing in the domain. So it doesn't say that there are *exactly* two things in the domain. Rather, it says that there are *no more than* two things in the domain.

No More Than Three

We could go on. For instance, the following wff says that there are no more than three things in the domain.

$$(\exists x)(\exists y)(\exists z)(\forall w)((w = x \vee w = y) \vee w = z)$$

In English, this says that there are three things, x , y , and z , such that every thing in the domain is either identical to x , or it's identical to y , or it's identical to z . Again, this could be true if there's only one or two things in the domain (in that case, either $x = y$ or $y = z$ or $x = z$). So it doesn't say that there are *exactly* three things in the domain. But it does say that there are no more than three.

No More Than One P

If we wish to say that there is no more than one thing *which is P* , then we may say that there is some thing such that, if anything is P , then that thing is it.

$$(\exists x)(\forall y)(Py \supset y = x)$$

Note that this doesn't entail that *something* is P . The above wff is true in an interpretation in which *nothing* is P (since, no matter what we let y refer to, ' Py ' will be false, so the conditional ' $Py \supset y = x$ ' will be true). What it does guarantee is that *either* nothing is P , *or* exactly one thing is P . That is: it guarantees that *no more* than one thing is P .

No More Than Two P s

In a similar fashion, if we wish to say that there is no more than two things which are P , then we may say that there are two things such that, if anything is P , then that thing is identical to one of them.

$$(\exists x)(\exists y)(\forall z)(Pz \supset (z = x \vee z = y))$$

Again, this doesn't guarantee that there are any P s; the above wff is true if nothing is P . It is also true if there is only one thing which is P ; and it is true if there are two things that are P . It is only false if there are three or more things which are P . For suppose that there are three things which are P —call them ' α ', ' β ', and ' γ '. Then, if x or y refer to anything other than α , β , or γ , then ' $(\forall z)(Pz \supset (z = x \vee z = y))$ ' will be false, since we may let z refer to α , and the antecedent will be true but the consequent will be false. So the conditional will not be true no matter what z refers to. So the universally quantified claim ' $(\forall z)(Pz \supset (z = x \vee z = y))$ ' will be false. So x and y must both refer to either α , β , or γ . But then, whichever one is left over (whichever one neither x nor y refer to), let z refer to that, and then ' $Pz \supset (z = x \vee z = y)$ ' will be false. So ' $(\forall z)(Pz \supset (z = x \vee z = y))$ ' will be false. So, if there are three or more things which are P , then ' $(\exists x)(\exists y)(\forall z)(Pz \supset (z = x \vee z = y))$ ' will be false.

No More Than Three P s

We could go on. If we wish to say that there is no more than three things which are P , then we may say that there are *three* things such that, if anything is P , then that thing is identical to one of them.

$$(\exists x)(\exists y)(\exists z)(\forall w)(Pw \supset ((w = x \vee w = y) \vee w = z))$$

Exactly One

Suppose that we wish to put *both* an upper bound *and* a lower bound on the number of things in the domain. In the case of one, saying that there is exactly one thing in the domain is equivalent to saying that there is no more than one thing in the domain (since we required that our domains have at least one thing in them). And we have already seen that the way to say that there is no more than one thing in the domain is:

$$(\exists x)(\forall y)y = x$$

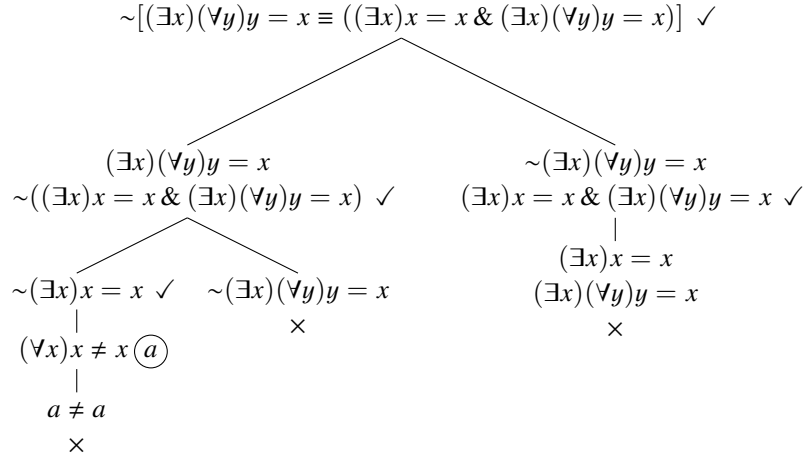


Figure 6.3: : The *PLI* tree establishing that ‘ $(\exists x)(\forall y)y = x$ ’ and ‘ $(\exists x)x = x \ \& \ (\exists x)(\forall y)y = x$ ’ are *PTI*-equivalent.

However, we could also just conjoin the claim that there is at least one thing with the claim that there is no more than one thing in the domain, as follows:

$$(\exists x)x = x \ \& \ (\exists x)(\forall y)y = x$$

These two wffs are *PLI*-equivalent, and *PTI*-equivalent, as the *PLI* tree shown in figure 6.3 establishes.

Exactly Two

Suppose that we wish to say that there are *exactly two* things in the domain. Then, we may just conjoin the claims that there are at least two things with the claim that there are no more than two things, as follows:

$$(\exists x)(\exists y)x \neq y \ \& \ (\exists x)(\exists y)(\forall z)(z = x \vee z = y)$$

This works, but it’s a bit more complicated than it needs to be. We may alternatively just say that there are two things which are non-identical, and that anything else in the domain is identical to one of them:

$$(\exists x)(\exists y)(x \neq y \ \& \ (\forall z)(z = x \vee z = y))$$

These two claims are *PLI* equivalent. The tree which establishes this equivalence is incredibly complicated, but I’ve reproduced it for you in figure 6.4.

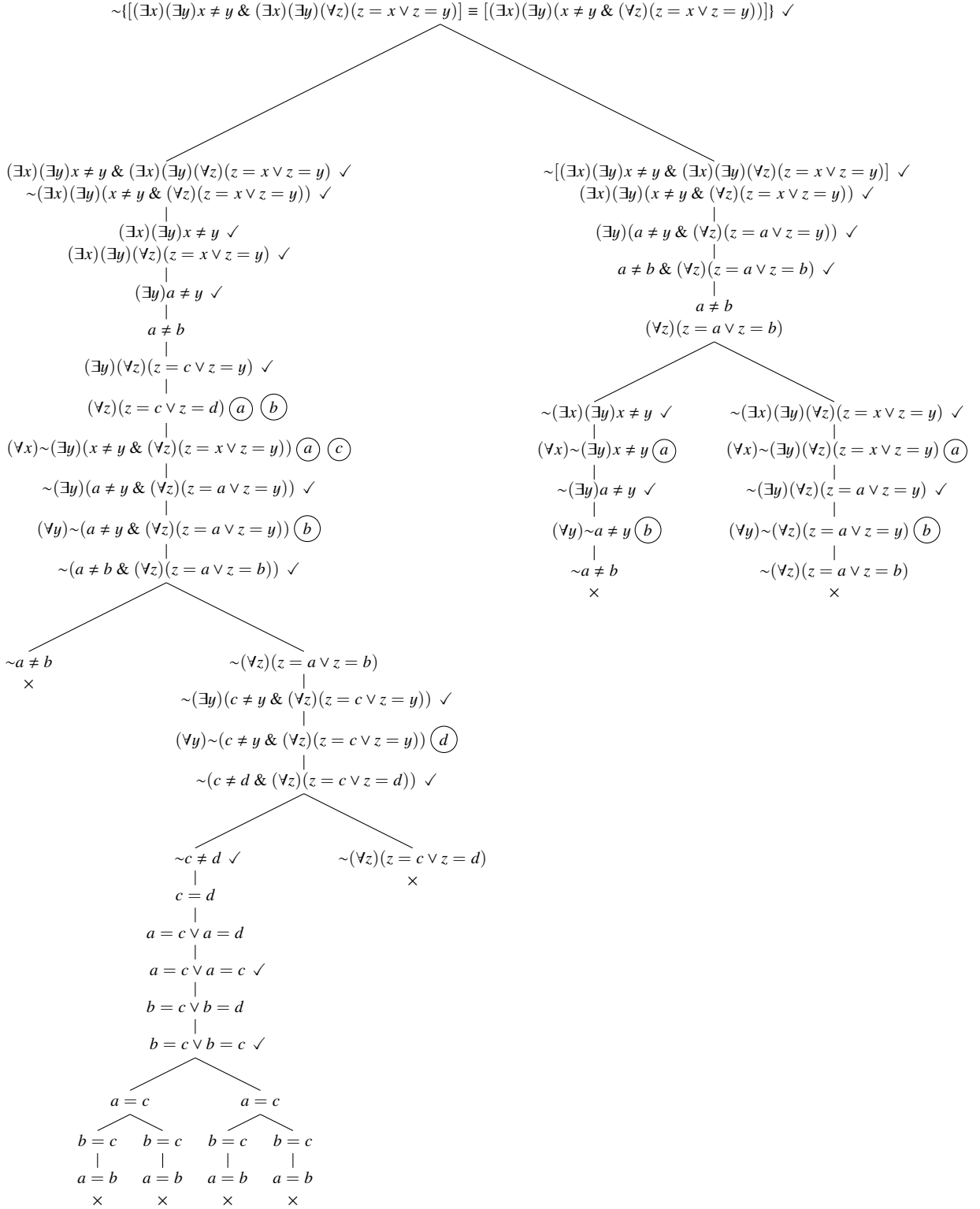


Figure 6.4: : The *PLI* tree which establishes that ‘ $(\exists x)(\exists y)x \neq y \ \& \ (\exists x)(\exists y)(\forall z)(z = x \vee z = y)$ ’ and ‘ $(\exists x)(\exists y)(x \neq y \ \& \ (\forall z)(z = x \vee z = y))$ ’ are *PTI*-equivalent.

Exactly Three

We could go further. Here's a way of saying that there are exactly three things in the domain.

$$(\exists x)(\exists y)(\exists z)((x \neq y \ \& \ y \neq z) \ \& \ z \neq x) \ \& \ (\forall w)((w = x \vee w = y) \vee w = z)$$

This wff says: 1) there are three things; 2) those things are distinct; and 3) everything in the domain is identical to one of them. And this tells us that there are exactly three things in the domain.

Exactly One P

To say that there is exactly one thing which is P , we may say that there is something which is P , and then add that anything *else* that's P must be identical to it:

$$(\exists x)(Px \ \& \ (\forall y)(Py \supset y = x))$$

Exactly Two P s

Similarly, to say that there are exactly two distinct things which are P , we may say that there are two things which are P , and then add that anything *else* that's P must be identical to one of them.

$$(\exists x)(\exists y)(x \neq y \ \& \ (Px \ \& \ (Py \ \& \ (\forall z)(Pz \supset (z = x \vee z = y))))))$$

Exactly Three P s

Similarly, to say that there are exactly three things which are P , we say that there are three distinct things which are P , and then add that anything else which is P must be identical to one of them.

$$(\exists x)(\exists y)(\exists z)(x \neq y \ \& \ (x \neq z \ \& \ (y \neq z \ \& \ (Px \ \& \ (Py \ \& \ (Pz \ \& \ (\forall w)(Pw \supset (w = x \vee (w = y \vee w = z))))))))))$$

6.10 The Only

Suppose that we wish to translate claims like

James is the only person in the class taller than 7 feet.

If we have the following partial *PLI*-interpretation,

$$\mathcal{I}^P = \begin{cases} \mathcal{D} & = \text{people in the class} \\ j & = \text{James} \\ T\mathbf{x} & = \mathbf{x} \text{ is taller than 7 feet} \end{cases}$$

then we may translate this sentence of English by saying two things. First, we must say that James is taller than 7 feet. This part of the translation is easy:

$$Tj$$

Then, we must say that he is the *only* one taller than 7 feet. We can accomplish this within *PLI* by saying that all people who are taller than 7 feet are identical to James.

$$(\forall x)(Tx \supset x = j)$$

Thus, the following sentence will translate “James is the only person in the class taller than 7 feet”:

$$Tj \& (\forall x)(Tx \supset x = j)$$

In general, we may translate claims of the form “*a* is the only *P*” as follows:

$$Pa \& (\forall x)(Px \supset x = a)$$

Suppose that we wish to translate “Only Bob and Eric went to the party”, given the following partial *PLI*-interpretation:

$$\mathcal{I}_p = \begin{cases} \mathcal{D} & = \text{contextually salient people} \\ b & = \text{Bob} \\ e & = \text{Eric} \\ P\mathbf{x} & = \mathbf{x} \text{ went to the party} \end{cases}$$

We may do this as follows:

$$(Pb \& Pe) \& (\forall y)(Py \supset (y = b \vee y = e))$$

6.11 Definite Descriptions

Suppose that we wish to translate a sentence like

The King of France is bald.

In order to attempt to translate this claim, we should think about what it's saying. It appears to be saying:

1. There is a King of France (the *existence* claim);
2. He is the only King of France (the *uniqueness* claim); and
3. He is bald (the *predication* claim).

Our translation should include these three elements of the original claim. But we already know how to say these three things in *PLI*. Suppose that we have the following (partial) *PLI*-interpretation,

$$\mathcal{I}_p = \begin{cases} \mathcal{D} & = \text{the set of people} \\ K\mathbf{x} & = \mathbf{x} \text{ is King of France} \\ B\mathbf{x} & = \mathbf{x} \text{ is bald} \end{cases}$$

Then, I submit, "The King of France is bald" may be translated by the wff

$$(\exists x)((Kx \& (\forall y)(Ky \supset y = x)) \& Bx)$$

In this wff, the claim that there is a King of France (the *existence* claim) is made by:

$$(\exists x)((Kx \dots$$

The *uniqueness* claim is made by:

$$\dots (\forall y)(Ky \supset y = x) \dots$$

And the *predication* claim (that he is bald), is made by:

$$\dots Bx \dots$$

In general, if we wish to translate a definite description of the form

The **P** is **Q**.

Then we may do so with a wff of the following form:

$$(\exists x)((\mathbf{P}x \ \& \ (\forall y)(\mathbf{P}y \supset y = x)) \ \& \ \mathbf{Q}x)$$

For instance, suppose that we wish to translate

The owner is tired.

given the following (partial) *PLI*-interpretation:

$$\mathcal{I}^{\mathcal{P}} = \begin{cases} \mathcal{D} & = \text{the set of people under discussion} \\ \mathbf{O}\mathbf{x} & = \mathbf{x} \text{ is an owner} \\ \mathbf{T}\mathbf{x} & = \mathbf{x} \text{ is tired} \end{cases}$$

Then, we may do so by saying:

1. There is an owner: $(\exists x)((\mathbf{O}x \dots;$
2. He is the only owner: $\dots (\forall y)(\mathbf{O}y \supset y = x) \dots;$ and
3. He is tired: $\dots \mathbf{T}x \dots$

Thus, “The owner is tired” is translated by:

$$(\exists x)((\mathbf{O}x \ \& \ (\forall y)(\mathbf{O}y \supset y = x)) \ \& \ \mathbf{T}x)$$

Chapter 7

Metatheory for Sentence Logic

We have now learned all of the logical systems which we are going to learn in this course. In part, learning about these logical systems consisted in learning new languages—first, the language *SL*, then the language *PL*, and finally, the language *PLI*—and learning a theory about the logical relations between (sets of) the sentences (or wffs) of those languages. In this part of the course, we will begin thinking *about* the logical theories we've been learning up to this point. In order to do so clearly, it will be important that we get clear on some important distinctions. The first is the distinction between *using* an expression of language and merely *mentioning* it.

7.1 Use & Mention

Consider the following pair of sentences:

1. 'Zoë' is a name.
2. Zoë is a name.

The first sentence is true. It says of *the string of Arabic characters*, or *the word*, 'Zoë' that that string of characters or that word functions as a name in English. And this is true. The second sentence, however, is false. That sentence says of *the person* Zoë that she is a name. But that is false. *Zoë has* a name—'Zoë'—but she is not *herself* a name.

This points to an important distinction between what takes place when we *use* a piece of language to talk about something non-linguistic—for instance, when we use the name

‘Zoë’ to talk about the *person* Zoë—and what takes place when we *mention* a piece of language to talk about *it*—that is, to talk about that piece of language itself.

When we are being careful, we mark this distinction by placing single quotation marks around the piece of language that we wish to speak about. For instance, in sentence (1), we are not attempting to *use* the name ‘Zoë’ to talk about the person Zoë. Rather, we are merely *mentioning* the name ‘Zoë’ to talk about *it*. On the other hand, in the sentence (2), we are not talking about the word ‘Zoë’. Rather, we are merely *using* that word to talk about something non-linguistic: the person Zoë.

7.2 Object Language & Metalanguage

Sometimes, we may want to *use* one language to talk *about* another language. For instance, we may want to *use* English to talk *about* the language Spanish. We could, for instance, use the language English to say what a word of Spanish means, as when we say:

3. ‘Gato’ is the Spanish word for cat.

In this case, we say that the language we are talking *about* is the *object language*. And we say that the language we are *using* to talk about the object language is the *metalanguage*.

The OBJECT LANGUAGE is the language we are talking about.

The METALANGUAGE is the language we are using to talk about the object language.

Thus, in sentence (3) above, the object language is Spanish—it is the language we are talking about—and the metalanguage is English—it is the language we are using to talk about Spanish.

Which language is the object language and which is the metalanguage is not fixed in stone. It depends upon which language we happen to be using and which language we happen to be talking about (supposing that we are talking about a language at all—if we are not talking about any language, then there will be no object language and no metalanguage). For instance, I could use Spanish to talk about the language English, as in:

4. ‘Cat’ es la palabra inglesa por gato.

(In English, this sentence says that the English word ‘cat’ is the English word for cat.) In this example, the object language is English, and the metalanguage is Spanish.

In all of the examples we will be concerned with in this part of the course, we will be taking the object language to be one of the logical languages we have learned about—either *SL*, or *PL*, or *PLI*. And our metalanguage will be English—though we will add to English some additional expressions for talking about the wffs of our object language. For instance, one addition to English will be *metavariables*.

7.3 Metavariables

Throughout the course, I have been utilizing METAVARIABLES to talk about *arbitrary* pieces of the languages *SL*, *PL*, or *PLI*. I am calling these variables ‘metavariables’ because they are variables that I am using in the metalanguage—English—to talk *about* bits of language in the object language—*SL*, *PL*, or *PLI*. These are like the variables which show up in *PL* and *PLI*, except that they are not a part of the language *PL* or *PLI*. In general, a VARIABLE is something which can take on various *values*. For instance, in *PL*, a variable ‘*x*’ can take on as a value anything which is in the domain \mathcal{D} of our *PL*-interpretation. The variable *height (in cm)* is a variable which can take any positive number, depending on the height of the thing we are considering; so its values are positive real numbers.

For METAVARIABLES, the potential values will be different bits of the object language. To review, we have been using the following conventions regarding metavariables:

1. Boldface uppercase letters like ‘**P**’, ‘**Q**’ and ‘**R**’ are used as metavariables whose potential values are well-formed formulae of the logical language we are interested in.
2. Uppercase script letters like ‘ \mathcal{F} ’ or ‘ \mathcal{H} ’ are used as metavariables whose potential values are predicates of *PL* or *PLI*.
3. Boldface lowercase letters from ‘a’—‘e’, like ‘**a**’, ‘**b**’, and ‘**c**’ are used as metavariables whose potential values are *constants* of *PL* or *PLI*.
4. Boldface lowercase ‘**f**’, ‘**g**’, and ‘**h**’ are used as metavariables whose potential values are *function symbols* of *PLI*.
5. Boldface lowercase ‘**t**’ (perhaps with subscripts) is used as a metavariable whose potential values are *terms* of *PL* or *PLI*.
6. Boldface lowercase ‘**x**’, ‘**y**’, ‘**z**’, or ‘**w**’ are used as metavariables whose potential values are *variables* of *PL* or *PLI*.

To these established conventions, let me add one more:

7. Uppercase Greek characters like ‘ Γ ’ and ‘ Δ ’ are used as metavariables whose values are *sets* of well-formed formulae of the language language we are interested in.

7.3.1 Use and Mention with Metavariables

Think back to the definition of *well-formed formulae* I provided for *SL*. Suppose that I want to say that, if you have two strings of characters, both of which are wffs, and you stick an ampersand between them and enclose the result in parentheses, then what you get is a wff. Here’s one attempt at saying that:

If ‘ \mathbf{P} ’ is a wff of *SL*, then ‘ $\sim\mathbf{P}$ ’ is a wff of *SL*.

But this doesn’t say what I wanted to say. In this, when I write “ \mathbf{P} ”, I am *mentioning* the metavariable ‘ \mathbf{P} ’. This metavariable is a part of the metalanguage. So what it says is that, if the *metavariable* ‘ \mathbf{P} ’ is a wff of *SL*, then... But I wasn’t trying to talk about a piece of the metalanguage. I was trying to talk about a piece of the object language.

Here’s a trick to get around this trouble: I may introduce a kind of quotation mark that I’ve already been using for the entire course: this is known as *corner quotation*, or *Quine quotation*, after W.V.O. Quine, who first introduced it. A corner quoted expression ‘ $\ulcorner\mathbf{P}\urcorner$ ’ works like this: it *mentions* the bit of language which is the *value* of the metavariable ‘ \mathbf{P} ’. Thus if \mathbf{P} takes the value ‘ $(A \supset F)$ ’, then ‘ $\ulcorner\mathbf{P}\urcorner$ ’ refers to ‘ $(A \supset F)$ ’.

More generally, expressions like ‘ $\ulcorner(\mathbf{P} \ \& \ \mathbf{Q})\urcorner$ ’ refer to the piece of language that you get when you concatenate an open parenthesis ‘(’ with the *value* of the metavariable ‘ \mathbf{P} ’, with ‘&’, with the *value* of the metavariable ‘ \mathbf{Q} ’ with a closed parenthesis ‘)’. So, for instance, if the metavariable ‘ \mathbf{P} ’ takes the value ‘ $(A \vee B)$ ’ and the metavariable ‘ \mathbf{Q} ’ takes the value ‘ $(C \equiv D)$ ’, then ‘ $\ulcorner(\mathbf{P} \ \& \ \mathbf{Q})\urcorner$ ’ refers to ‘ $(A \vee B) \ \& \ (C \equiv D)$ ’.

Consider the following claim (which is true):

$\ulcorner\mathbf{P}\urcorner$ is a name for $\ulcorner\mathbf{P}\urcorner$.

To think about what this is saying, think about what will happen if we substitute some expression, say ‘ $(A \vee \sim B)$ ’, for the metavariable ‘ \mathbf{P} ’. If we do this, then each pair of corner quotes will turn into regular quotes, and we will get the following:

“($A \vee \sim B$)” is a name for ‘ $(A \vee \sim B)$ ’.

This says that the *quoted* expression “ $(A \vee \sim B)$ ” is a name for the wff ‘ $(A \vee \sim B)$ ’. And this is true. (In the first part of the sentence, I *mention* the quoted expression to talk about *it*—the singly-quoted expression—and in the second part of the sentence, I *use* the singly quoted expression to talk about the unquoted wff of *SL*.)

On the other hand, consider the following claim (which is false):

‘ $\ulcorner \mathbf{P} \urcorner$ ’ is a name for $\ulcorner \mathbf{P} \urcorner$.

Here, ‘ $\ulcorner \mathbf{P} \urcorner$ ’ refers to the concatenation of the symbols ‘ \ulcorner ’, ‘ \mathbf{P} ’, and ‘ \urcorner ’. It is therefore talking about a bit of the metalanguage. To think about what the claim is saying, think about what will happen if we substitute some expression, say ‘ $(A \vee \sim B)$ ’, for the metavariable ‘ \mathbf{P} ’:

‘ $\ulcorner \mathbf{P} \urcorner$ ’ is a name for ‘ $(A \vee \sim B)$ ’.

While the second set of corner quotes change to regular quotes, the first pair of corner quotes do not, since they are not being *used*; rather, they are being *mentioned*. The subject of the sentence is a name for the sequence of symbols: ‘ \ulcorner ’, followed by ‘ \mathbf{P} ’, followed by ‘ \urcorner ’. And this is not a name for any wff of *SL*. It is a name for a sentence of the *metalanguage*, not a sentence of the *object language*.

7.4 Syntax and Semantics

When we learned about our three languages, *SL*, *PL*, and *PLI*, we were careful to distinguish between those parts of the language that had to do merely with the *form* of the wffs of the language—what we called the ‘syntax’ of the language—and the parts of the language that told us something about the *meaning* of the wffs of the language—what we called the ‘semantics’ of the language.

In general, *syntactic* features are features of the language that have nothing to do with the meaning or the interpretation of the symbols appearing in the language. In order to know that ‘ $(A \supset \sim B)$ ’ is a wff of *SL*, you didn’t have to know anything about what ‘ $(A \supset \sim B)$ ’ *means*. Whether this is a wff is an entirely syntactic matter. It is just a matter of the symbols that appear and the order they appear in.

To make this point clearer, let’s consider the following language, which we can call ‘*KL*’ (for ‘Kooky Language’). The vocabulary of *KL* consists of the following symbols:

1. Shapes:

$\square, \circ, \triangle$

2. Parentheses (,)

And we can specify which ways of stringing together these symbols counts as a *well-formed formula* with the following definition:

1. '○' is a wff of *KL*.
2. If 'P' is a wff of *KL*, then '□P□' is a wff of *KL*.
3. If 'P' is a wff of *KL*, then '△P△' is a wff of *KL*.
4. Nothing else is a wff of *KL*.

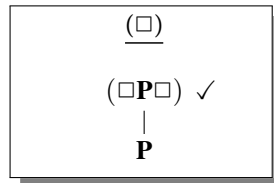
Now, we can say whether '□(△○△)□' is a wff of *KL* (it is), and we can say whether '□□□' is a wff of *KL* (it is not). And we can say all of this without having any idea about the meaning of either of these wffs (in fact, they have no meaning at all; I didn't define any semantics for this language—it's all syntax).

SYNTACTIC features are features of expressions that can be understood without knowing anything about the *meaning* of those expressions.

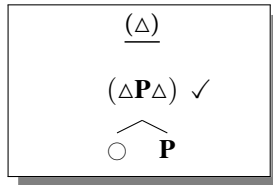
SEMANTIC features are features of expressions that can only be understood if you know something about the *meaning* of those expressions.

Note, now, that whether a given *SL* tree or *PL* tree closes is an entirely syntactic matter. Given an *SL* or *PL* tree, we can figure out whether that tree abides by the rules for trees without knowing anything about what the expressions appearing in that tree *mean*.

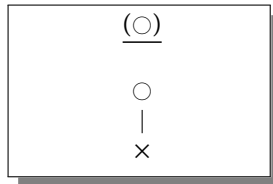
To make that clearer, we could define up tree rules for our kooky language *KL*.



This rule says “if you have a wff of the form '□P□', then you may, at any point, check that wff off and extend every open branch on which the wff lies by writing down 'P' at the bottom of that branch”.

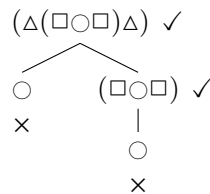


This rule says “if you have a wff of the form $\ulcorner (\Delta P \Delta) \urcorner$, then you may, at any point, check that wff off and split every open branch on which the wff lies by writing down $\ulcorner \circ \urcorner$ on the left-hand-side and writing down $\ulcorner P \urcorner$ on the right-hand side.”



This rule says: “If, at any point, a circle appears by itself on a branch, then close that branch by writing ‘×’ at the bottom of that tree.”

With these rules, we can go on to say whether a tree is correct and complete in a way precisely similar to the way that we did in *SL*. For instance, the following tree, beginning with $\ulcorner (\Delta(\Box\Box\Box)\Delta) \urcorner$ at its root, is correct and complete



We can know all this without knowing anything at all about the *meaning* of the expressions showing up on this tree (after all, these expressions don't *have* any meaning!). Whether the tree is correct and complete, and whether it closes, is entirely a matter of whether it conforms to purely syntactic rules. And precisely the same thing goes for the trees of *SL*, *PL*, and *PLI*.

Whether a tree closes is an entirely *syntactic* question. In order to answer this question, we need not know anything about what the wffs appearing on the tree *mean*—*i.e.*, we need not know anything about the conditions under which the wffs are true or false.

On the other hand, whether an argument is *valid*, or whether a set of wffs is *consistent*, is *not* a purely syntactic question. In order to answer the question of whether an argument of *SL* is *SL*-valid, we must know something about the conditions under which those wffs are *true* and *false*.

For instance, we are not in a position to say whether the *KL*-argument

$$\circ // (\Box\Box\Box)$$

is *KL*-valid or *KL*-invalid until we know something about the circumstances in which the wffs of *KL* are true or false. And we are not in a position to say whether the set of wffs of *KL*

$$\{(\Delta(\Box\Box\Box)\Delta), (\Box\Box\Box), \circ\}$$

is *KL*-consistent or *KL*-inconsistent until we know what these wffs *mean*.

Whether an argument is valid is *not* a purely syntactic question. In order to answer this question, we must know something about the *meaning* of the wffs of *SL*.

7.5 Syntactic Validity and Semantic Validity

The central question we will be concerned with during this part of the course is whether the purely syntactic rules we have set up for the trees of *SL* (and *PL*) are good ones. That is: whether what they tell us about whether an argument of *SL* (or *PL*) is valid is correct.

To have a convenient shorthand for discussing this question, let's introduce a bit of notation. In the first place, we will introduce a convenient shorthand for the claim that an argument of *SL* with the set of premises Γ and the conclusion $\ulcorner \mathbf{P} \urcorner$ is *ST*-valid or *ST*-invalid. If (and only if) the argument with the set of premises Γ and the conclusion $\ulcorner \mathbf{P} \urcorner$ is *ST*-valid, we will write $\ulcorner \Gamma \vdash_{ST} \mathbf{P} \urcorner$.

$\ulcorner \Gamma \vdash_{ST} \mathbf{P} \urcorner$ means that every correct *SL* tree beginning with all and only the wffs in Γ and $\ulcorner \sim \mathbf{P} \urcorner$ at its root closes.

If (and only if) the argument with the set of premises Γ and the conclusion $\ulcorner \mathbf{P} \urcorner$ is *SL*-valid, we will write $\ulcorner \Gamma \models_{SL} \mathbf{P} \urcorner$

$\ulcorner \Gamma \models_{SL} \mathbf{P} \urcorner$ means that there is no truth-value assignment which makes every wff in Γ true and makes $\ulcorner \mathbf{P} \urcorner$ false.

And similarly, to make the claim that every PL tree with the wffs in Γ and $\lceil \sim \mathbf{P} \rceil$ at its root closes, I will write

$$\Gamma \vdash_{PL} \mathbf{P}$$

And to make the claim that the argument with the premise set Γ and the conclusion $\lceil \mathbf{P} \rceil$ is PL -valid, I will write

$$\Gamma \models_{PL} \mathbf{P}$$

$\lceil \Gamma \vdash_{PL} \mathbf{P} \rceil$ means that every correct PL tree beginning with all and only the wffs in Γ and $\lceil \sim \mathbf{P} \rceil$ at its root closes.

$\lceil \Gamma \models_{PL} \mathbf{P} \rceil$ means that there is no PL -interpretation which makes every wff in Γ true and makes $\lceil \mathbf{P} \rceil$ false.

If it is clear from context which logical system we are talking about (or if I wish to talk about *any* logical system without specifying one specifically), then I will omit the subscripted ' SL ' or ' PL ', and just write

$$\Gamma \vdash \mathbf{P}$$

for the claim that the argument with the premise set Γ and the conclusion \mathbf{P} is *syntactically* valid (*i.e.*, tree-valid); and I will sometimes similarly just write

$$\Gamma \models \mathbf{P}$$

for the claim that the argument with the premise set Γ and the conclusion \mathbf{P} is *semantically* valid.

7.6 Soundness and Completeness of the Tree Method

The point of this part of the course is to prove that the tree method for testing for validity is a good one. However, there are two ways that the tree method could be good (and two corresponding ways that it could be bad) that it will be useful to clearly separate. In the first place, it could be that, whenever the tree method tells us that an argument is valid, that argument *is* valid. That is, it could be that, whenever a tree beginning with the premises in Γ and $\lceil \sim \mathbf{P} \rceil$ at its root closes, the argument from the premises in Γ to the conclusion $\lceil \mathbf{P} \rceil$ is semantically valid. If this is so, then we will say that the tree method is *sound*.

SOUNDNESS The tree method is sound iff, for any set Γ and wff $\ulcorner \mathbf{P} \urcorner$:

$$\text{if } \Gamma \vdash \mathbf{P} \text{ then } \Gamma \models \mathbf{P}$$

If the tree method is sound, then this means that we can always trust what it has to say when it tells us that an argument is valid. However, it does not automatically follow that we can always trust what it has to say about *invalidity*. For instance, suppose that we have a syntactic account of validity which tells us that only one argument of *SL* is valid: $A \& B // A$. For every other argument of *SL*, the syntactic account says that it is invalid. Such an account would be *sound*—we could always trust it if it tells us that an argument is valid, since $A \& B // A$ is an *SL*-valid argument. However, we could not always trust it when it told us that an argument is *invalid*. It would, for instance, tell us that the argument $A \& B // B$ is *invalid*; but this is incorrect. That argument is valid. So even though we could trust what this account tells us about *validity*, we could not trust what it tells us about *invalidity*.

We will say that the tree method is *complete* if we can trust what it has to say about *invalidity*. That is, we will say that the tree method is *complete* iff, whenever the tree method says that an argument is invalid, it *is* invalid. That is: the tree method is complete iff, whenever a tree beginning with the premises in Γ and $\ulcorner \sim \mathbf{P} \urcorner$ at its root remains open, the argument from the premises in Γ to the conclusion $\ulcorner \mathbf{P} \urcorner$ is semantically invalid.

COMPLETENESS The tree method is complete iff, for any set Γ and wff $\ulcorner \mathbf{P} \urcorner$:

$$\text{if } \Gamma \models \mathbf{P} \text{ then } \Gamma \vdash \mathbf{P}$$

Note that, just because the tree method is *complete*, this doesn't mean that the tree method is *sound*. We could have an account of validity that told us that every argument was valid *except* for $A \supset B // B \supset A$, which it told us was invalid. Now, we could trust what this account has to tell us about *invalidity*, since $A \supset B // B \supset A$ is invalid. However, we could not trust what it has to tell us about validity, since this account would tell us that $A \supset B // B$ is valid, which is false.

So neither soundness or completeness is sufficient for the other. If an account of validity is sound *and* complete, then we can trust everything that it has to say about validity and we can trust everything that it has to say about *invalidity*. So if the tree method is both sound and complete, then we can trust it, no matter what it tells us about the validity or invalidity of an argument. It will tell us that an argument is valid if and only if the argument is valid. The goal of this portion of the class will be to prove that the tree method is both sound and complete.

Recall: our goal in this section of the course is to establish that an argument of *SL* is *SL*-valid if and only if it is *ST*-valid (and similarly for *PL*-validity and *PT*-validity). That

is: we wish to show that the tree method will tell us that an argument is valid when and only when that argument really is valid. Using the notation from last time, we may say that the argument from a set of premises Γ to the conclusion \mathbf{P} is *SL*-valid by writing ' $\Gamma \models_{SL} \mathbf{P}$ ', and we may say that the tree beginning with only the wffs in Γ and $\sim\mathbf{P}$ at its root closes by writing ' $\Gamma \vdash_{SL} \mathbf{P}$ '.

Then, we saw that we could split the claim we wish to prove up into two separate parts:

Soundness. *If an argument is ST-valid, then it is SL-valid.*

$$\text{If } \Gamma \vdash_{ST} \mathbf{P} \text{ then } \Gamma \models_{SL} \mathbf{P}$$

Completeness. *If an argument is SL-valid, then it is ST-valid.*

$$\text{If } \Gamma \models_{SL} \mathbf{P} \text{ then } \Gamma \vdash_{ST} \mathbf{P}$$

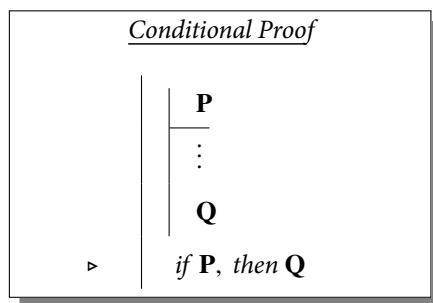
Establishing these two claims is our goal in this section of the course. In order to accomplish this goal, we will need to make use of some logical methods of reasoning.

For instance, we will make use of logical inference rules like *modus ponens*.

<u>Modus Ponens</u>	
	<i>if P, then Q</i>
	P
▷	Q

This rule says the following: if you know that *if P, then Q*, and you know that **P**, then you may infer that **Q**. We know from *SL* that such a method of inference will never take us from truth to falsehood (that is: we know that the method of inference is *valid*). So we will feel free to use it when we offer our proofs of soundness and completeness.

For another example, we will need to make use of the method of *conditional proof*, which some of you may have encountered in PHIL 0500:



This rule says the following: if you *assume* that **P**, and from this assumption, you are able to conclude that **Q**, then you may conclude (outside of the scope of any assumption) that the conditional *if P, then Q* is true.

However, we will need more than simple logical methods of reasoning like this. We will also need a proof technique known as *mathematical induction*.

7.7 Mathematical Induction

Suppose that you have an infinite number of dominoes, all arranged in a line. The only thing you know about these dominoes are the following:

1. The first one falls.
2. If a domino falls, then the domino which comes after it falls, too.

Can you work out whether all of the dominoes will fall? It looks like you can. For you know that the first domino falls. And you know that, if the first domino falls, then the second domino falls as well. So, by *modus ponens*, the second domino falls. So now you know that the second domino falls. And you know that, if the second domino falls, then the third domino falls too. So, by *modus ponens*, you know that the third domino falls. So you know that the third domino falls. And you know that, if the third domino falls, then the fourth domino falls too. So, by *modus ponens*, then fourth domino falls. So you know that the fourth domino falls. And you know that, if the fourth domino falls, then the fifth domino falls too...

We can continue reasoning in this way forever, without end. If we do so, then we will have proven that every domino falls. So, from claims like (1) and (2) above, we may work out that every domino falls.

Now that we know that, from claims like (1) and (2), it follows that every domino falls, it's not really important that we go through the steps of reasoning it through that every

domino will fall, given (1) and (2). We could simply look at (1) and (2) and conclude straight away that every domino will fall. And if we were able to establish that (1) and (2) are true, then we would have *already* established that it is true that every domino falls.

This is the method of mathematical induction. In general in mathematical induction, we have some infinite list of things, and we wish to show that every thing in our infinite list has some property. In order to do that, we show two things:

1. The first item on the list has the property.
2. If an item on the list has the property, then so does the item after it on the list.

Just like with the dominoes, once we've shown these two claims, we will have *already* shown that every item on the list has the property. For we could just run through the very same kind of reasoning that we went through above with respect to the dominoes. From (1), we know that the first item on the list has the property. And, from (2), we know that, if the first item on the list has the property, then the second item on the list has the property, too. So, by *modus ponens*, we know that the second item on the list has the property. And, from (2), we know that, if the second item on the list has the property, then the third item on the list has the property, too. So, by *modus ponens*, we know that the third item on the list has the property... And we could just go on reasoning in this fashion, forever, without end. And, if we did so, then we would have established that every item on the list has the property.

7.7.1 Terminology

That's really all that there is to mathematical induction. The only thing left to learn about it are some common bits of terminology that get used when discussing it.

In the first place, we have a given infinite list. We say that this list is the list *on which we are doing the induction*. Since the list may change from mathematical induction to mathematical induction, the list on which we are doing the induction may change from induction to induction.

We then have some property that we are trying to show that every item on the list has. This property is known as the **INDUCTIVE PROPERTY**. The inductive property is the property that we are trying to show that every item on the list has.

We show that every item on the list has the inductive property by showing two things: firstly, that the first item on the list has the property; and secondly, that, for any k , if the k th item on the list has the property, then the $k + 1$ st item on the list has the property

too. The first thing we show is known as THE BASIS STEP. The second thing we show is known as THE INDUCTIVE STEP.

1. The BASIS STEP establishes that the first item on the list has the inductive property.
2. The INDUCTIVE STEP establishes that, for any k , if the k th item on the list has the inductive property, then the $k + 1$ st item on the list has the inductive property, too.

As we've seen above, if we prove the basis step and the inductive step, then we will have proven that every item on the list has the inductive property.

Proving the basis step is usually not that complicated. However, we should say something about how to prove the inductive step. The way we will prove the inductive step is by using *conditional proof*.

The Inductive Step

<div style="border-right: 1px solid black; padding-right: 10px; margin-right: 10px;"> <p>the kth item on the list has the inductive property.</p> <p style="text-align: center;">⋮</p> <p>the $k + 1$st item on the list has the inductive property.</p> </div>	<p>► For any k, if the kth item on the list has the property,</p> <p>then the $k + 1$st item on the list has the property</p>
---	--

That is, we will *assume* that the k th item on the list has the inductive property; and, from this assumption, we will prove that the $k + 1$ st item on the list has the inductive property. This assumption that we will make is known as the INDUCTIVE HYPOTHESIS. That is: the inductive hypothesis is that the k th item on the list has the inductive property.

In the inductive hypothesis, we assume that, for *an arbitrary* number k , the k th item on the list has the property. In order for our assumption to be about an *arbitrary* number k , we must not assume anything at all about the number k . If, from the assumption that the k th item on the list has the property, we can show that the $k + 1$ st item on the list has the property—for an *arbitrary* k —then we will have shown that *for any* k , if the k th item on the list has the property, then the $k + 1$ st item on the list has the property, too. We will have shown this because, if we do not assume anything at all about the number k , then we know that our conditional proof will work for *any* number k .

7.7.2 Examples of Mathematical Induction

Suppose that we wish to prove the following claim:

Claim 1. *For every counting number n , the result of adding all counting numbers less than or equal to n is $n \cdot (n + 1)/2$.*

$$1 + 2 + 3 + \cdots + n = \frac{n \cdot (n + 1)}{2}$$

We may establish this claim by using mathematical induction. To see how to do so, notice that the claim we wish to establish has the logical form of

$$(\forall n)Pn$$

That is: for any number n , n has a certain property, P . This property P is the inductive property. It is the property of the sum of all numbers less than or equal to n being equal to n times $n + 1$ divided by 2.

$$Pn \stackrel{\text{def}}{=} 1 + \cdots + n = \frac{n \cdot (n + 1)}{2}$$

To establish our claim by mathematical induction, we must first think about what the list is that we are doing induction on. That list is just the list of all of the natural numbers. That is, we are doing induction on the list 1, 2, 3, 4, 5, 6, 7, ...

To begin with, we must first establish the basis step. That is, we must show that the first item on our list has the property.

Basis Step. *If $n = 1$, then*

$$\frac{n \cdot (n + 1)}{2} = \frac{1 \cdot (1 + 1)}{2} = \frac{2}{2} = 1$$

So, the claim holds if $n = 1$.

Next, we must prove the inductive step. We do so by making an inductive hypothesis: that, for an arbitrary k , k has the inductive property. From this assumption, we derive the assumption that $k + 1$ has the property as well.

Inductive Step. *Assume, for the purposes of conditional proof, the inductive hypothesis:*

Inductive Hypothesis. *For an arbitrary number k ,*

$$1 + 2 + 3 + \cdots + k = \frac{k \cdot (k + 1)}{2}$$

Then, if we add $k + 1$ to both the left and right hand sides of the equation above, we get

$$\begin{aligned}
 1 + 2 + 3 + \cdots + k + (k + 1) &= \frac{k \cdot (k + 1)}{2} + (k + 1) \\
 &= \frac{k \cdot (k + 1)}{2} + \frac{2(k + 1)}{2} \\
 &= \frac{k \cdot (k + 1) + 2(k + 1)}{2} \\
 &= \frac{(k + 1)(k + 2)}{2} \\
 &= \frac{(k + 1)[(k + 1) + 1]}{2}
 \end{aligned}$$

Thus, the claim holds of $k + 1$

So, from the assumption of the **inductive hypothesis** that the claim holds of an arbitrary k , we can show that the claim holds of $k + 1$ as well. Thus, *for any* k , if the claim holds of k , then the claim holds of $k + 1$ as well.

In this course, we will be interested in using mathematical induction to prove things about *SL* and *PL*. So let us look at an example involving those languages.

Suppose that we wish to establish the following claim:

Claim 2. *Every wff of SL which has no operators other than ‘~’ is an SL-contingency.*

To begin with, we must find some way of *enumerating* the wffs of *SL* which have no operators other than \sim . While we could enumerate all of the wffs of *SL* individually, it will be simpler if we instead look at *sets* of wffs of *SL*. In the first set are all the statement letters of *SL*. In the second set are all of the negated statement letters of *SL*. In the third are all of the negated negated statement letters of *SL*. And so on. That is, the list of sets of wffs on which we will do our induction is the following:

$$\begin{array}{l}
 \text{set 1) } \{A, \quad B, \quad C, \quad \dots\} \\
 \text{set 2) } \{\sim A, \quad \sim B, \quad \sim C, \quad \dots\} \\
 \text{set 3) } \{\sim \sim A, \quad \sim \sim B, \quad \sim \sim C, \quad \dots\} \\
 \text{set 4) } \{\sim \sim \sim A, \quad \sim \sim \sim B, \quad \sim \sim \sim C, \quad \dots\} \\
 \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \ddots
 \end{array}$$

And the inductive property we will be interested in is the property of *containing only SL-contingencies*. That is, the first item on our list (the set containing all of the statement letters of *SL*) has the property iff it contains only *SL*-contingencies. This is the basis step.

Basis Step. Every wff of SL in the first set is a statement letter of SL . A truth-value assignment is an assignment of truth-value to the statement letters of SL . For every statement letter, there is some truth-value assignment on which it is true and some truth-value assignment on which it is false. Therefore, every statement letter of SL is an SL -contingency. So the first set of wffs of SL contains only SL -contingencies.

Next, we must prove the inductive step. For this, we will begin by *assuming*, for an arbitrary k , that all of the wffs in the k th set are SL -contingencies, and then showing that, if this assumption is true, then we may show that all of the wffs in the $k + 1$ st set are SL -contingencies also.

Inductive Step. Assume, for the purposes of conditional proof, the *inductive hypothesis*.

Inductive Hypothesis. All of the wffs in the k th set are SL -contingencies.

Every wff in the $k + 1$ st set is the result of negating a wff in the k th set. Thus, every wff $\lceil P \rceil$ in the $k + 1$ st set is of the form

$$\sim Q$$

for some $\lceil Q \rceil$ in the k th set. By the inductive hypothesis, every $\lceil Q \rceil$ in the k th set is an SL -contingency. So there are some truth-value assignments on which $\lceil Q \rceil$ is true and some truth-value assignments on which $\lceil Q \rceil$ is false. On every truth-value assignment on which $\lceil Q \rceil$ is true, $\lceil \sim Q \rceil$ is false (by the definition of ' \sim '). And on every truth-value assignment on which $\lceil Q \rceil$ is false, $\lceil \sim Q \rceil$ is true (by the definition of ' \sim '). Thus, for every $\lceil P \rceil$ in the $k + 1$ st set, there are some truth-value assignments on which $\lceil P \rceil$ is true and some truth-value assignments on which $\lceil P \rceil$ is false. Thus, every $\lceil P \rceil$ in the $k + 1$ st set is an SL -contingency.

From the assumption that every wff in the k th set is an SL -contingency, we were able to show that every wff in the $k + 1$ st set is an SL -contingency too. k was arbitrary, so we may conclude that for any k , if every wff in the k th set is an SL -contingency, then every wff in the $k + 1$ st set is an SL -contingency also.

7.8 Varieties of Mathematical Induction

We've seen the basics of the proof technique of mathematical induction. However, there are several nuances we should spend some time looking at. There are a few different varieties of mathematical induction, which are distinguished by the form of the inductive hypothesis. The variety of mathematical induction we have just been looking at is known as WEAK INDUCTION.

WEAK INDUCTION In weak induction, the inductive hypothesis is that, for some arbi-

bitrary k , the k th item on our list has the inductive property. From this, we show that the $k + 1$ st item on our list has the inductive property.

For some applications of mathematical induction, it will be simpler to make another kind of inductive hypothesis. Following Teller, we may call this the **STRONG FORM OF WEAK INDUCTION**:

STRONG FORM OF WEAK INDUCTION In the strong form of weak induction, the inductive hypothesis is that, for some arbitrary k , every item on our list up to the k th has the inductive property. From this, we show that the $k + 1$ st item on our list has the inductive property.

That is, when we are using the strong form of weak induction, we assume that, for all numbers $i \leq k$, the i th item on our list has the inductive property. And we then go on to show that the $k + 1$ st item on our list has the inductive property as well.

If we have completed the basis step of an induction, by showing that:

The first item on our list has the inductive property.

$$\mathcal{P}(1)$$

And we have completed the inductive step by showing that the following universally quantified conditional is true:

For any k , if all items on our list up to and including k have the inductive property, then the $k + 1$ st item on our list has the inductive property, too.

$$(\forall i)(i \leq k \supset \mathcal{P}(i)) \supset \mathcal{P}(k + 1)$$

Then we may reason as follows: we know that the first item on our list has the inductive property. So, every item up to and including the first item has the inductive property. So, by the inductive step, the second item on our list must have the inductive property, too. So the first and the second items on our list have the inductive property. So every item up to and including the second item on our list have the inductive property. By the inductive step, the third item on our list must have the inductive property too. So every item up to and including the third item on our list has the inductive property. So, by the inductive step, the fourth item on our list must have the inductive property, too. So every item up to and including the fourth item on our list has the inductive property. By the inductive step...

If we continue reasoning in this way, then we will eventually cover every item on the list. The reasoning at play here is precisely similar to the reasoning at play in the form of mathematical induction we considered before. However, this alternate formulation will prove more useful with certain problems.

7.9 Choosing the Right Inductive Property

Consider the following claim:

Claim 3. *Every wff of SL which has ‘&’ as its only logical operator is not an SL -contradiction (that is: it is true on some truth-value assignment).*

If we wish to prove this claim with mathematical induction, then we will first need to construct some list of the wffs of SL with ‘&’ as their only logical operators. This is the list that we will attempt to do mathematical induction on. It looks like the following list will do nicely:

- set 0) The set containing all of the statement letters of SL .
- set 1) The set of all wffs of SL with ‘&’ as their only operator which contain 1 occurrence of ‘&’.
- set 2) The set of all wffs of SL with ‘&’ as their only operator which contain 2 occurrences of ‘&’.
- set 3) The set of all wffs of SL with ‘&’ as their only operator which contain 3 occurrences of ‘&’.
- ⋮
- set n) The set of all wffs of SL with ‘&’ as their only operator which contain n occurrences of ‘&’.
- ⋮

Which inductive property should we choose? Well, we wish to show that every item on this list contains only wffs which are not SL -contradictions, so it’s natural to choose the following inductive property:

INDUCTIVE PROPERTY A set on our list has the inductive property iff that set contains no SL -contradictions.

However, it turns out that completing the inductive step with this inductive property will be rather difficult. Let's see why.

When we make the inductive hypothesis, we should grant ourselves the assumption that every set on our list *up to and including* set k contains no *SL*-contradictions, and attempt to show that from this assumption we can show that set $k + 1$ contains no *SL*-contradictions.

Inductive Hypothesis. *For some arbitrary k , all sets up to and including set k contain no *SL*-contradictions.*

Consider now how the inductive step might go. We might *try* to say the following (*note: this is not a valid proof of the inductive step*).

This is not a valid proof of the inductive step!!!

Inductive Step. *Every wff in the set k is of the form $\lceil \mathbf{P} \ \& \ \mathbf{Q} \rceil$, with wffs $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ belonging to sets $1-k$. By the inductive hypothesis, neither $\lceil \mathbf{P} \rceil$ nor $\lceil \mathbf{Q} \rceil$ are *SL*-contradictions. Thus, there is some truth-value assignment on which $\lceil \mathbf{P} \rceil$ is true and some truth-value assignment on which $\lceil \mathbf{Q} \rceil$ is true. Therefore, there is some truth-value assignment on which $\lceil \mathbf{P} \ \& \ \mathbf{Q} \rceil$ is true. Therefore, $\lceil \mathbf{P} \ \& \ \mathbf{Q} \rceil$ is not an *SL*-contradiction.*

This is not a valid proof of the inductive step!!!

Why is this not a valid proof? Because, simply because $\lceil \mathbf{P} \rceil$ is true on some truth-value assignment and $\lceil \mathbf{Q} \rceil$ is true on some truth-value assignment, this *does not* mean that $\lceil \mathbf{P} \ \& \ \mathbf{Q} \rceil$ is true on some truth-value assignment.

Let $\lceil \mathbf{P} \rceil = \lceil A \rceil$, and let $\lceil \mathbf{Q} \rceil = \lceil \sim A \rceil$. Then, while $\lceil \mathbf{P} \rceil$ is true on some truth-value assignment and $\lceil \mathbf{Q} \rceil$ is true on some truth-value assignment, $\lceil \mathbf{P} \ \& \ \mathbf{Q} \rceil$ is not true on any truth-value assignment. $\lceil A \ \& \ \sim A \rceil$ is an *SL*-contradiction.

So it turns out that proving the inductive step is hard if we try to prove with with the following inductive property *containing no *SL*-contradictions*.

We may make our work easier by choosing to prove something *stronger* than the thing that we ultimately wish to show.¹ Rather than showing that none of the sets on our list contain any *SL*-contradictions, we could instead attempt to show that all of the wffs in each set on our list has the following property: it is true on the truth-value assignment which makes each statement letter true. That is, we may choose the following inductive property:

¹ One claim is *stronger* than another iff the first entails the second, but the second doesn't entail the first.

INDUCTIVE PROPERTY A set on our list has the inductive property iff that set contains only wffs which are true on the truth-value assignment which makes every statement letter of SL true.

It is rather easy to show that the first set on our list has this property:

Basis Step. *Set 0 contains only statement letters of SL . Trivially, all of these statement letters are true on the truth-value assignment which makes all of the statement letters of SL true.*

And it is similarly easy to complete the inductive step.

Inductive Step. *Assume, for the purposes of conditional proof, that the inductive hypothesis is true.*

Inductive Hypothesis. *For some arbitrary k , all sets up to and including set k contain only wffs which are true on the truth-value assignment which makes every statement letter of SL true.*

Every wff in set $k + 1$ is of the form

$$(P \ \& \ Q)$$

for some $\ulcorner P \urcorner$ and $\ulcorner Q \urcorner$ which appear in sets 0 – k . By the inductive hypothesis, then, both $\ulcorner P \urcorner$ and $\ulcorner Q \urcorner$ are true on the truth-value assignment which makes all statement letters of SL true. Therefore, by the definition of ‘&’ (a conjunction is true iff both of its conjuncts are true), $\ulcorner (P \ \& \ Q) \urcorner$ is also true on the truth-value assignment which makes all statement letters true. Since $\ulcorner P \urcorner$ and $\ulcorner Q \urcorner$ were arbitrary, every wff in set $k + 1$ is true on the truth-value assignment which makes every statement letter of SL true.

By conditional proof, for any k , if all sets up to and including set k contain only wffs which are true on the truth-value assignment which makes all of the statement letters of SL true, then set $k + 1$ contains only wffs which are true on the truth-value assignment which makes all of the statement letters of SL true.

The lesson: we should be shrewd in our choice of inductive property. Sometimes, it makes sense to choose an inductive property which is more informative than the property which we are trying to show that every item on our list possesses.

7.10 More Examples of Mathematical Induction

Suppose that we wish to establish the following claim:

Claim 4. *Any wff of SL which contains only the operator ‘ \vee ’ is not an SL -contradiction.*

To begin to prove this claim using mathematical induction, we must find some way of enumerating the wffs of SL which contain only ‘ \vee ’. Here is one natural way of doing it:

- set 0) The set of all statement letters of SL .
- set 1) The set of all wffs of SL containing only the operator ‘ \vee ’ which contain one occurrence of ‘ \vee ’.
- set 2) The set of all wffs of SL containing only the operator ‘ \vee ’ which contain two occurrences of ‘ \vee ’.
- set 3) The set of all wffs of SL containing only the operator ‘ \vee ’ which contain three occurrences of ‘ \vee ’.
- ⋮
- set n) The set of all wffs of SL containing only the operator ‘ \vee ’ which contain n occurrences of ‘ \vee ’.

Every wff of SL which contains only the operator ‘ \vee ’ will appear within one of the sets on this list.

Next, we must decide upon an inductive property which we wish to show every item on our list has. The natural choice of inductive property is the property of *containing no SL -contradictions*. Let’s ride with this inductive property and see if things work out.

We may complete the basis step as follows:

Basis Step. *Every statement letter of SL has some truth-value assignment on which it is true. Thus, no statement letter of SL is an SL -contradiction. So no wff in set 0 is an SL -contradiction. So set 0 contains no SL -contradictions.*

Now, suppose that we attempted to complete the inductive step by making the following inductive hypothesis:

Inductive Hypothesis. *For an arbitrary k , set k contains no SL -contradictions.*

The problem is that, if we use the list above, we could not use this inductive hypothesis to show that the set $k + 1$ contains no SL contradictions. For example, suppose that we are thinking about set 2—that is, suppose that $k = 2$. Now, the inductive hypothesis tells us that all wffs from set 2 are not SL -contradictions. And we wish to show that set 3 does not contain any SL -contradictions. Set 3 contains the following wff of SL :

$$((A \vee B) \vee (C \vee D))$$

This is a disjunction whose two disjuncts are ' $(A \vee B)$ ' and ' $(C \vee D)$ '. But *neither of these wffs are from set 2*. Both of these wffs are from set 1. So our inductive hypothesis doesn't tell us that they are not *SL*-contradictions.

We may get around this problem by using the strong form of weak induction. That is: we may complete the inductive step by making the following inductive hypothesis:

Inductive Step. *Assume, for the purposes of conditional proof, the following inductive hypothesis:*

Inductive Hypothesis. *For some arbitrary k , all sets up to and including set k contain no *SL*-contradictions.*

Then, every wff in set $k + 1$ is of the form

$$(\mathbf{P} \vee \mathbf{Q})$$

*For some wffs ' \mathbf{P} ', ' \mathbf{Q} ' from sets 1– k . Since ' \mathbf{P} ' is from sets 1– k , by the inductive hypothesis, there is some truth-value assignment which makes ' \mathbf{P} ' true. But any truth-value assignment which makes ' \mathbf{P} ' true makes ' $(\mathbf{P} \vee \mathbf{Q})$ ' true. So, there is some truth-value assignment which makes ' $\mathbf{P} \vee \mathbf{Q}$ ' true. So ' $(\mathbf{P} \vee \mathbf{Q})$ ' is not an *SL*-contradiction. Since ' \mathbf{P} ' and ' \mathbf{Q} ' were arbitrary, the same goes for every other wff in set $k + 1$. So set $k + 1$ contains no *SL*-contradictions.*

7.10.1 Number of Parentheses

Let's use mathematical induction to prove the following claim:

Claim 5. *For every wff of *SL*, ' \mathbf{P} ', if ' \mathbf{P} ' has n binary operators—where ' \vee ', ' $\&$ ', ' \supset ', and ' \equiv ' are binary operators—then ' \mathbf{P} ' has $2n$ parentheses.*

First, let's think through what this says. If we have a wff with 0 binary operators, like, *e.g.*,

$$\begin{aligned} & D \\ & \sim \sim A \\ & \sim \sim \sim F \end{aligned}$$

this wffs will have $2 \cdot 0 = 0$ parentheses. And if we have a wff with 1 binary operator,

like, *e.g.*,

$$\begin{aligned} &(E \vee P) \\ &\sim (K \supset L) \\ &(\sim \sim H \equiv \sim Z) \end{aligned}$$

this wff will have $2 \cdot 1 = 2$ parentheses. And if we have a wff with 2 binary operators, like, *e.g.*,

$$\begin{aligned} &((E \vee P) \supset \sim L) \\ &(\sim (K \supset L) \vee H) \\ &((\sim \sim H \equiv \sim Z) \& Y) \end{aligned}$$

this wff will have $2 \cdot 2 = 4$ parentheses.

The claim seems to hold true when we look at these example wffs. We wish to show that it holds in general, of any wff of *SL*. Intuitively, the reason that it seems to hold is that, every time we introduce a new binary operator, we write down two extra parentheses; so the total number of parentheses should just be two times the number of binary operators. We can show this more rigorously using mathematical induction. Here's the way we'll do it: we'll start with an infinitely long list such that every wff of *SL* shows up at some point on the list. Since we want to show a property that has to do with the number of binary operators in the wff, the following seems like a good choice:

- set 0) The set containing all wffs of *SL* with 0 binary operators.
- set 1) The set containing all wffs of *SL* with 1 binary operator.
- set 2) The set containing all wffs of *SL* with 2 binary operators.
- set 3) The set containing all wffs of *SL* with 3 binary operators.
- ⋮
- set *n*) The set containing all wffs of *SL* with *n* binary operators.

The inductive property we wish to show that every item on this list has is the following:

INDUCTIVE PROPERTY A set on our list has the inductive property iff every wff in that set it has twice as many parentheses as it has binary operators.

Start with the basis step:

Basis Step. Every wff with 0 parentheses is either a statement letter or some number of tildes in front of a statement letter. Statement letters don't have parentheses, and negations of negations of...statement letters don't have parentheses either. So every wff in set 0 will have $2 \cdot 0 = 0$ parentheses. So every wff in set 0 has twice as many parentheses as it has binary operators.

Then, the inductive step:

Inductive Step. Assume, for the purposes of conditional proof, the *inductive hypothesis*.

Inductive Hypothesis. For some arbitrary k , every set up to and including set k is such that the members of that set have twice as many parentheses as they have binary operators.

Then, every wff in set $k + 1$, $\lceil \mathbf{P} \rceil$, will have one of the following forms:

1. $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$
2. $\lceil (\mathbf{Q} \& \mathbf{R}) \rceil$
3. $\lceil (\mathbf{Q} \supset \mathbf{R}) \rceil$
4. $\lceil (\mathbf{Q} \equiv \mathbf{R}) \rceil$
5. $\lceil \dots \sim (\mathbf{Q} \vee \mathbf{R}) \rceil$
6. $\lceil \dots \sim (\mathbf{Q} \& \mathbf{R}) \rceil$
7. $\lceil \dots \sim (\mathbf{Q} \supset \mathbf{R}) \rceil$
8. $\lceil \dots \sim (\mathbf{Q} \equiv \mathbf{R}) \rceil$

for some $\lceil \mathbf{Q} \rceil$, $\lceil \mathbf{R} \rceil$ from sets 0– k .

Let $\lceil \#_{op}(\mathbf{Q}) \rceil$ denote the number of binary operators in $\lceil \mathbf{Q} \rceil$, and let $\lceil \#_{op}(\mathbf{R}) \rceil$ denote the number of binary operators in $\lceil \mathbf{R} \rceil$. And let $\lceil \#_{par}(\mathbf{Q}) \rceil$ denote the number of parentheses in $\lceil \mathbf{Q} \rceil$ and similarly let $\lceil \#_{par}(\mathbf{R}) \rceil$ denote the number of parentheses in $\lceil \mathbf{R} \rceil$.

If $\lceil \mathbf{P} \rceil$ is of the form $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$, $\lceil (\mathbf{Q} \& \mathbf{R}) \rceil$, $\lceil (\mathbf{Q} \supset \mathbf{R}) \rceil$, $\lceil (\mathbf{Q} \equiv \mathbf{R}) \rceil$, $\lceil \dots \sim (\mathbf{Q} \vee \mathbf{R}) \rceil$, $\lceil \dots \sim (\mathbf{Q} \& \mathbf{R}) \rceil$, $\lceil \dots \sim (\mathbf{Q} \supset \mathbf{R}) \rceil$, or $\lceil \dots \sim (\mathbf{Q} \equiv \mathbf{R}) \rceil$, then, since $\lceil \mathbf{P} \rceil$ has $k + 1$ binary operators, $\lceil \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ must have a total of k binary operators between the two of them. That is:

$$\#_{op}(\mathbf{Q}) + \#_{op}(\mathbf{R}) = k \quad (\star)$$

Both $\lceil \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ are from sets 0– k . So, by the inductive hypothesis,

$$\begin{aligned} \#_{par}(\mathbf{Q}) &= 2\#_{op}(\mathbf{Q}) & \#_{par}(\mathbf{R}) &= 2\#_{op}(\mathbf{R}) \\ \frac{1}{2}\#_{par}(\mathbf{Q}) &= \#_{op}(\mathbf{Q}) & \frac{1}{2}\#_{par}(\mathbf{R}) &= \#_{op}(\mathbf{R}) \end{aligned}$$

Putting this together with (\star), we get:

$$\begin{aligned}\frac{1}{2}\#_{par}(\mathbf{Q}) + \frac{1}{2}\#_{par}(\mathbf{R}) &= k \\ \frac{1}{2}(\#_{par}(\mathbf{Q}) + \#_{par}(\mathbf{R})) &= k \\ \#_{par}(\mathbf{Q}) + \#_{par}(\mathbf{R}) &= 2k\end{aligned}$$

Then, since $\lceil \mathbf{P} \rceil$ is of the form $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$, $\lceil (\mathbf{Q} \& \mathbf{R}) \rceil$, $\lceil (\mathbf{Q} \supset \mathbf{R}) \rceil$, $\lceil (\mathbf{Q} \equiv \mathbf{R}) \rceil$, $\lceil \dots \sim (\mathbf{Q} \vee \mathbf{R}) \rceil$, $\lceil \dots \sim (\mathbf{Q} \& \mathbf{R}) \rceil$, $\lceil \dots \sim (\mathbf{Q} \supset \mathbf{R}) \rceil$, or $\lceil \dots \sim (\mathbf{Q} \equiv \mathbf{R}) \rceil$, the number of parentheses in $\lceil \mathbf{P} \rceil$ is the number of parentheses in $\lceil \mathbf{Q} \rceil$ plus the number of parentheses in $\lceil \mathbf{R} \rceil$, plus 2.

$$\begin{aligned}\#_{par}(\mathbf{P}) &= \#_{par}(\mathbf{Q}) + \#_{par}(\mathbf{R}) + 2 \\ &= 2k + 2 \\ &= 2(k + 1)\end{aligned}$$

So, $\lceil \mathbf{P} \rceil$ has twice as many parentheses as binary operators.

From the inductive hypothesis that, for an arbitrary k , every wff with k or fewer binary operators has $2k$ parentheses, we were able to show that every wff with $k + 1$ binary operators has $2(k + 1)$ binary operators.

Since k was arbitrary, we may conclude that, for any k , if every wff with k or fewer binary operators has $2k$ parentheses, then every wff with $k + 1$ binary operators has $2(k + 1)$ binary operators.

7.10.2 Substitution of SL-equivalents

Claim 6. For any wff of SL $\lceil \mathbf{P} \rceil$, if $\lceil \mathbf{Q} \rceil$ is a subformula of $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q}^* \rceil$ is SL-equivalent to $\lceil \mathbf{Q} \rceil$, then the result of replacing $\lceil \mathbf{Q} \rceil$ with $\lceil \mathbf{Q}^* \rceil$ in $\lceil \mathbf{P} \rceil$ is SL-equivalent to $\lceil \mathbf{P} \rceil$.

For example, $\lceil (A \vee B) \rceil$ is a subformula of $\lceil ((A \vee B) \supset C) \rceil$, and $\lceil (B \vee A) \rceil$ is SL-equivalent to $\lceil (A \vee B) \rceil$. So, the claim tells us that $\lceil ((B \vee A) \supset C) \rceil$ is SL-equivalent to $\lceil ((A \vee B) \supset C) \rceil$.

To complete the induction, we must first specify which list of things we are doing the induction on. The following list is, in general, a good one to use if you wish to show that all the wffs of SL have a certain property:

- set 0) The set of all statement letters of SL.
- set 1) The set of all wffs of SL with one logical operator.

set 2) The set of all wffs of SL with two logical operators.

set 3) The set of all wffs of SL with three logical operators.

⋮

set n) The set of all wffs of SL with n logical operators.

⋮

Then, a wff like

$$(\sim A \supset \sim B)$$

will appear in set 3, since it has three logical operators: it has one ‘ \sim ’, one ‘ \supset ’, and another ‘ \sim ’. And a wff like

$$(\sim(A \& B) \supset \sim(D \equiv F))$$

will appear in set 5, since it has 5 logical operators: two ‘ \sim ’s, one ‘ $\&$ ’, one ‘ \supset ’, and one ‘ \equiv ’.

We next need to choose an inductive property. The thing we wish to show about the wffs of SL is that, for any wff ‘ \mathbf{P} ’, for every ‘ \mathbf{P}^* ’ which is the result of replacing some subformula of ‘ \mathbf{P} ’ with an SL equivalent subformula, ‘ \mathbf{P}^* ’ is SL equivalent to ‘ \mathbf{P} ’. This is a mouthful. To make it a bit simpler to say, let’s introduce a phrase—“equivalent under equivalent substitutions”—to stand for this property.

A wff of SL ‘ \mathbf{P} ’ is EQUIVALENT UNDER EQUIVALENT SUBSTITUTIONS iff, for every wff ‘ \mathbf{P}^* ’ which is the result of replacing some subformula of ‘ \mathbf{P} ’ with an SL equivalent subformula, ‘ \mathbf{P}^* ’ is SL -equivalent to ‘ \mathbf{P} ’.

Then, our inductive property (the property we are trying to show that every item on our list has), is the property of containing only wffs which are equivalent under equivalent substitutions.

For any i , set i has the inductive property iff set i contains only wffs which are equivalent under equivalent substitutions.

First, we must complete the basis step

Basis Step. *Every wff in set 0 is a statement letter. Since every wff is a subformula of itself, every statement letter has one and only one subformula: itself. If we replace a statement letter with a wff which is SL -equivalent to it, then what we will get is trivially a wff which is SL -equivalent to the statement letter. So set 0 contains only wffs which are equivalent under equivalent substitutions.*

Next, for the inductive step, we assume, in our inductive hypothesis, that every set up to and including set k contains only wffs which are equivalent under equivalent substitutions, and then use this assumption to conclude that set $k + 1$ contains only wffs which are equivalent under equivalent substitutions.

Inductive Step. Assume, for the purposes of conditional proof, the **inductive hypothesis**

Inductive Hypothesis. For some arbitrary k , every set up to and including set k contains only wffs which are equivalent under equivalent substitutions.

Then, every wff $\lceil \mathbf{P} \rceil$ in set $k + 1$ will have one of the following forms

1. $\lceil \sim \mathbf{Q} \rceil$
2. $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$
3. $\lceil (\mathbf{Q} \& \mathbf{R}) \rceil$
4. $\lceil (\mathbf{Q} \supset \mathbf{R}) \rceil$
5. $\lceil (\mathbf{Q} \equiv \mathbf{R}) \rceil$

for some wffs $\lceil \mathbf{Q} \rceil$, $\lceil \mathbf{R} \rceil$ from sets 0– k . By the inductive hypothesis, we know that both $\lceil \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ are equivalent under equivalent substitutions.

Trivially, if we substitute, for $\lceil \mathbf{P} \rceil$, a wff which is SL-equivalent to $\lceil \mathbf{P} \rceil$, then what we get will be SL-equivalent to $\lceil \mathbf{P} \rceil$.

Every other subformula of $\lceil \mathbf{P} \rceil$, however, will be a subformula of either $\lceil \mathbf{Q} \rceil$ or $\lceil \mathbf{R} \rceil$. We know, by the inductive hypothesis, that $\lceil \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ are equivalent under SL-substitutions. Thus, for any $\lceil \mathbf{Q}^* \rceil$ which is the result of substituting some subformula of $\lceil \mathbf{Q} \rceil$ with an SL-equivalent subformula, $\lceil \mathbf{Q}^* \rceil$ is SL-equivalent to $\lceil \mathbf{Q} \rceil$. And, for any $\lceil \mathbf{R}^* \rceil$ which is the result of substituting some subformula of $\lceil \mathbf{R} \rceil$ with an SL-equivalent subformula, $\lceil \mathbf{R}^* \rceil$ is SL-equivalent to $\lceil \mathbf{R} \rceil$. Thus, for any such $\lceil \mathbf{Q}^* \rceil$ or $\lceil \mathbf{R}^* \rceil$, $\lceil \mathbf{Q}^* \rceil$ will be true on all and only the truth-value assignments on which $\lceil \mathbf{Q} \rceil$ is true, and $\lceil \mathbf{R}^* \rceil$ will be true on all and only the truth-value assignments on which $\lceil \mathbf{R} \rceil$ is true.

Therefore, $\lceil \mathbf{P}^* \rceil$ will be SL-equivalent to $\lceil \mathbf{P} \rceil$. To make this even clearer, we can walk through the 5 cases above and see that, if you substitute a subformula of either $\lceil \mathbf{Q} \rceil$ or $\lceil \mathbf{R} \rceil$ in $\lceil \mathbf{P} \rceil$ for an SL-equivalent subformula, then the result will be SL-equivalent to $\lceil \mathbf{P} \rceil$. Since we know that $\lceil \mathbf{Q} \rceil$ and $\lceil \mathbf{Q}^* \rceil$ are SL-equivalent by the inductive hypothesis, they must be true on all the same truth-value assignments. And similarly for $\lceil \mathbf{R} \rceil$ and $\lceil \mathbf{R}^* \rceil$. Thus,

\mathbf{Q}	\mathbf{Q}^*	$\sim \mathbf{Q}$	$\sim \mathbf{Q}^*$
T	T	F	F
F	F	T	T

Q	Q*	R	R*	Q ∨ R	Q* ∨ R	Q ∨ R*
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

Q	Q*	R	R*	Q & R	Q* & R	Q & R*
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

Q	Q*	R	R*	Q ⊃ R	Q* ⊃ R	Q ⊃ R*
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>

Q	Q*	R	R*	Q ≡ R	Q* ≡ R	Q ≡ R*
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>

So, whether $\lceil \mathbf{P} \rceil$ is of the form $\lceil \sim \mathbf{Q} \rceil$, $\lceil \mathbf{Q} \vee \mathbf{R} \rceil$, $\lceil \mathbf{Q} \& \mathbf{R} \rceil$, $\lceil \mathbf{Q} \supset \mathbf{R} \rceil$, or $\lceil \mathbf{Q} \equiv \mathbf{R} \rceil$, $\lceil \mathbf{P} \rceil$ will be equivalent under equivalent substitutions.

But, since $\lceil \mathbf{P} \rceil$ was arbitrary, we may conclude that, for any wff $\lceil \mathbf{P} \rceil$ in set $k + 1$, $\lceil \mathbf{P} \rceil$ is equivalent under equivalent substitutions. Thus, set $k + 1$ contains only wffs which are equivalent under equivalent substitutions.

Thus, from the inductive hypothesis that all sets up to and including some arbitrary set k contain only wffs which are equivalent under equivalent substitutions, we may conclude that set $k + 1$ contains only wffs which are equivalent under equivalent substitutions.

Therefore, for any k , if all sets up to and including set k contain only wffs which are equivalent under equivalent substitutions, then set $k + 1$ contains only wffs which are equivalent

under equivalent substitutions.

7.10.3 Duals

Claim 7. For any wff of SL $\ulcorner \mathbf{P} \urcorner$ which contains only the logical operators ‘ \vee ’, ‘ $\&$ ’, and ‘ \sim ’, the wff that we get when place a ‘ \sim ’ in front of every statement letter appearing in $\ulcorner \mathbf{P} \urcorner$, replace every occurrence of ‘ \vee ’ with ‘ $\&$ ’, and replace every occurrence of ‘ $\&$ ’ with ‘ \vee ’—call this the dual of $\ulcorner \mathbf{P} \urcorner$ —is SL -equivalent to $\ulcorner \sim \mathbf{P} \urcorner$.

For example, if $\ulcorner \mathbf{P} \urcorner = (A \vee (B \& \sim C))$, then the dual of $\ulcorner \mathbf{P} \urcorner$ is $(\sim A \& (\sim B \vee \sim \sim C))$.

To prove this claim, we must first decide upon the list on which we are going to do the induction. The following is a natural choice:

- set 0) The set of all statement letters of SL .
- set 1) The set of all wffs of SL containing only ‘ \vee ’, ‘ $\&$ ’, and ‘ \sim ’ which have one logical operator.
- set 2) The set of all wffs of SL containing only ‘ \vee ’, ‘ $\&$ ’, and ‘ \sim ’ which have two logical operators.
- set 3) The set of all wffs of SL containing only ‘ \vee ’, ‘ $\&$ ’, and ‘ \sim ’ which have three logical operators.
- ⋮
- set n) The set of all wffs of SL containing only ‘ \vee ’, ‘ $\&$ ’, and ‘ \sim ’ which have n logical operators.

We next need to choose an inductive property. Let’s say, for any wff $\ulcorner \mathbf{P} \urcorner$, that $\ulcorner \mathbf{P} \urcorner$ is *dual-contradictory* iff the dual of $\ulcorner \mathbf{P} \urcorner$ is SL -equivalent to $\ulcorner \sim \mathbf{P} \urcorner$.

For any wff of SL with only the logical operators ‘ \vee ’, ‘ $\&$ ’, and ‘ \sim ’, $\ulcorner \mathbf{P} \urcorner$, $\ulcorner \mathbf{P} \urcorner$ is DUAL-CONTRADICTIONARY iff the dual of $\ulcorner \mathbf{P} \urcorner$, $dual(\ulcorner \mathbf{P} \urcorner)$, is SL -equivalent to $\ulcorner \sim \mathbf{P} \urcorner$.

Then, the inductive property—which we wish to show that every item on our list has—is the property of *containing only dual-contradictory wffs*.

Thus, for the basis step, we must show that every wff in set 0—that is, every statement letter—is dual-contradictory.

Basis Step. Set 0 contains all and only the statement letters of SL. For every statement letter $\lceil \mathbf{A} \rceil$, the dual of that statement letter is $\lceil \sim \mathbf{A} \rceil$. $\lceil \sim \mathbf{A} \rceil$ is trivially SL-equivalent to $\lceil \sim \mathbf{A} \rceil$. Thus, set 0 contains only dual-contradictory wffs.

Next, we need to establish the inductive step. To do so, we will assume the inductive hypothesis that, for some arbitrary k , all sets up to and including set k contain only dual-contradictory wffs. From the inductive hypothesis, we will show that set $k + 1$ contains only dual-contradictory wffs.

Inductive Step. Assume, for the purposes of conditional proof, the **inductive hypothesis**

Inductive Hypothesis. For some arbitrary k , all sets up to and including set k contain only dual-contradictory wffs.

Then, every wff $\lceil \mathbf{P} \rceil$ in set $k + 1$ will have one of the following forms

1. $\lceil \sim \mathbf{Q} \rceil$
2. $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$
3. $\lceil (\mathbf{Q} \& \mathbf{R}) \rceil$

For some $\lceil \mathbf{Q} \rceil$, $\lceil \mathbf{R} \rceil$ from sets 0– k . By the definition of ‘dual’,

1. $dual(\sim \mathbf{Q}) = \sim dual(\mathbf{Q})$
2. $dual((\mathbf{Q} \vee \mathbf{R})) = (dual(\mathbf{Q}) \& dual(\mathbf{R}))$
3. $dual((\mathbf{Q} \& \mathbf{R})) = (dual(\mathbf{Q}) \vee dual(\mathbf{R}))$

By the inductive hypothesis, we know that $dual(\mathbf{Q})$ is SL-equivalent to $\sim \mathbf{Q}$ and $dual(\mathbf{R})$ is SL-equivalent to $\sim \mathbf{R}$. That is:

\mathbf{Q}	\mathbf{R}	$dual(\mathbf{Q})$	$dual(\mathbf{R})$	$\sim \mathbf{Q}$	$\sim \mathbf{R}$
T	T	F	F	F	F
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

We may simply proceed by cases. If $\lceil \mathbf{P} \rceil = \lceil \sim \mathbf{Q} \rceil$, then $dual(\mathbf{P}) = \sim dual(\mathbf{Q})$, and

\mathbf{Q}	$dual(\mathbf{Q})$	$\sim \sim \mathbf{Q}$	$\sim dual(\mathbf{Q})$
T	F	T	T
F	T	F	F

Thus, $dual(\mathbf{P})$ is SL-equivalent to $\sim\mathbf{P}$.

If $\lceil \mathbf{P} \rceil = \lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$, then $dual(\mathbf{P}) = (dual(\mathbf{Q}) \& dual(\mathbf{R}))$, and

\mathbf{Q}	\mathbf{R}	$dual(\mathbf{Q})$	$dual(\mathbf{R})$	$\sim(\mathbf{Q} \vee \mathbf{R})$	$dual(\mathbf{Q}) \& dual(\mathbf{R})$
T	T	F	F	F	F
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

Thus, $dual(\mathbf{P})$ is SL-equivalent to $\sim\mathbf{P}$.

If $\lceil \mathbf{P} \rceil = \lceil (\mathbf{Q} \& \mathbf{R}) \rceil$, then $dual(\mathbf{P}) = (dual(\mathbf{Q}) \vee dual(\mathbf{R}))$, and

\mathbf{Q}	\mathbf{R}	$dual(\mathbf{Q})$	$dual(\mathbf{R})$	$\sim(\mathbf{Q} \& \mathbf{R})$	$dual(\mathbf{Q}) \vee dual(\mathbf{R})$
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	T	T

Thus, $dual(\mathbf{P})$ is SL-equivalent to $\sim\mathbf{P}$.

Therefore, $dual(\mathbf{P})$ is SL-equivalent to $\sim\mathbf{P}$ no matter whether $\lceil \mathbf{P} \rceil$ is of the form $\lceil \sim\mathbf{Q} \rceil$, $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$ or $\lceil (\mathbf{Q} \& \mathbf{R}) \rceil$.

Since $\lceil \mathbf{P} \rceil$ was arbitrary, the same holds for every $\lceil \mathbf{P} \rceil$ in set $k + 1$. Therefore, every wff in set $k + 1$ is dual-contradictory. So set $k + 1$ contains only dual-contradictory wffs.

Therefore, from the inductive hypothesis that all sets up to and including set k contain only dual-contradictory wffs, we were able to show that set $k + 1$ contains only dual-contradictory wffs.

Since k was arbitrary, for any k , if all sets up to and including set k contain only dual-contradictory wffs, then set $k + 1$ contains only dual contradictory wffs, too.

Chapter 8

Soundness of the Tree Method for Sentence Logic

Today we wish to show that the tree method for *SL* is *sound*. Recall, the tree method is sound iff we can trust everything that it has to say about *validity*—that is, iff, whenever the tree method says that an argument is valid, that argument is valid.

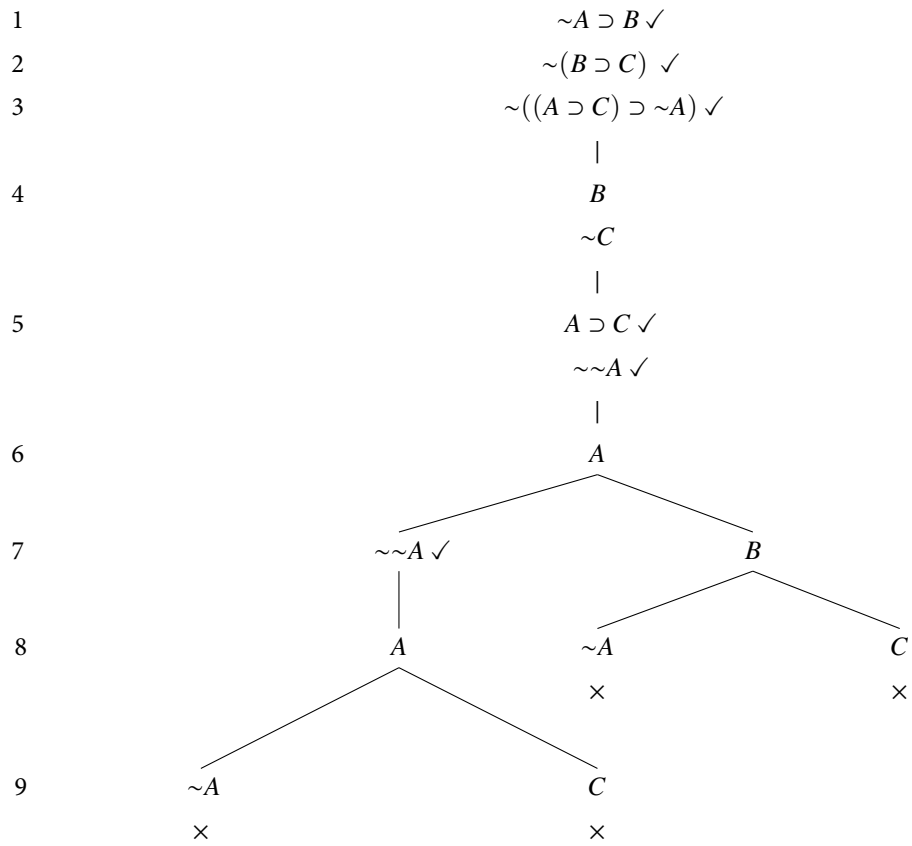
SOUNDNESS The tree method for *SL* is sound iff, for any set Γ and wff $\ulcorner \mathbf{P} \urcorner$:

$$\text{if } \Gamma \vdash_{SL} \mathbf{P} \text{ then } \Gamma \models_{SL} \mathbf{P}$$

8.1 Preliminaries

Before getting into the proof of soundness, I want to firm up some details of the truth trees. For the most part, I've been ignoring line numbering. However, for our proof that the tree method is sound, we will be doing mathematical induction on the lines of a tree, so it will be important that we're clear on how the lines get numbered, what it is for a wff to appear on a line, and what a branch is.

Consider the following tree.



This tree has 9 lines. Line 4 has two wffs written on it—one stacked on top of the other. That is: both ' B ' and ' $\sim C$ ' appear on line 4. Similarly, line 5 has two wffs written on it; both ' $A \supset C$ ' and ' $\sim\sim A$ ' are on line 5.

Line 7 also has two wffs written on it, but those wffs are on different branches. That is: ' $\sim\sim A$ ' and ' B ' are both written on line 7.

This tree has four overlapping branches. These four branches have the following wffs

written on them, in the following order.

	1st branch	2nd branch	3rd branch	4th branch
line 1	$\sim A \supset B$	$\sim A \supset B$	$\sim A \supset B$	$\sim A \supset B$
line 2	$\sim(B \supset C)$	$\sim(B \supset C)$	$\sim(B \supset C)$	$\sim(B \supset C)$
line 3	$\sim((A \supset C) \supset \sim A)$	$\sim((A \supset C) \supset \sim A)$	$\sim((A \supset C) \supset \sim A)$	$\sim((A \supset C) \supset \sim A)$
line 4	B	B	B	B
	$\sim C$	$\sim C$	$\sim C$	$\sim C$
line 5	$A \supset C$	$A \supset C$	$A \supset C$	$A \supset C$
	$\sim\sim A$	$\sim\sim A$	$\sim\sim A$	$\sim\sim A$
line 6	A	A	A	A
line 7	$\sim\sim A$	$\sim\sim A$	B	B
line 8	A	A	$\sim A$	C
line 9	$\sim A$	C		

Thus, ' $\sim A \supset B$ ', on line 1, appears on four different, overlapping branches. ' B ', on line 7, appears on two different, overlapping branches. ' $\sim A$ ', on line 8, appears on only one branch: the third one.

8.2 The Proof in Broad Outline

To begin with, let's appeal to a fact we have seen several times in the course thus far: an argument from the premises in Γ to the conclusion ' \mathbf{P} ' is *SL*-valid iff the set containing all of the premises in Γ and the negation of the conclusion ' $\sim\mathbf{P}$ ' is *SL*-inconsistent.

A bit of notation: if we have two sets Γ and Δ , then ' $\Gamma \cup \Delta$ ' refers to a set that has as members any member of *either* Γ or Δ .

$\Gamma \cup \Delta$ contains all things x such that either x is in Γ or x is in Δ

Then, another way of expressing the connection between consistency and validity is as follows:

$\Gamma \models_{SL} \mathbf{P}$ if and only if $\Gamma \cup \{\sim\mathbf{P}\}$ is *SL*-inconsistent

Thus, we could re-write the claim of SOUNDNESS as follows:

if $\Gamma \vdash_{SL} \mathbf{P}$ then $\Gamma \cup \{\sim\mathbf{P}\}$ is *SL*-inconsistent (SOUNDNESS)

Since $\lceil \mathbf{P} \supset \mathbf{Q} \rceil$ is equivalent to $\lceil \sim \mathbf{Q} \supset \sim \mathbf{P} \rceil$ (known as the *contrapositive* of $\lceil \mathbf{P} \supset \mathbf{Q} \rceil$), we may re-write the claim of SOUNDNESS by taking the contrapositive:

$$\text{if } \Gamma \cup \{\sim \mathbf{P}\} \text{ is } SL\text{-consistent then } \Gamma \not\vdash_{SL} \mathbf{P} \quad (\text{SOUNDNESS})$$

That is: if there *is* some truth-value assignment which makes all of the wffs in $\Gamma \cup \{\sim \mathbf{P}\}$ true, then any completed tree which begins with the wffs in $\Gamma \cup \{\sim \mathbf{P}\}$ at its root will have an open branch.

We will establish this claim by establishing something slightly more informative: *for any* finite set of wffs Δ , if there is some truth-value assignment which makes all of the wffs in Δ true, then any completed tree, Δ , which begins with the wffs in Δ at its root will have some branch such that there is some truth-value assignment makes all of the wffs appearing on that branch true.

To have a shorthand way of referring to this property, let's say that a branch is CONSISTENT iff there is some truth-value assignment which makes all of the wffs appearing on that branch true.

BRANCH CONSISTENCY A branch of a tree is CONSISTENT iff there is some truth-value assignment which makes all of the wffs appearing on that branch true.

Then, we will establish soundness by establishing the following claim:

Claim 8. *For any finite set of wffs of SL, Δ , if there is some truth-value assignment which makes every member of Δ true, then any tree \mathcal{T}_Δ which has the members of Δ at its root will have a consistent branch.*

Note that, in this claim, we are only talking about *finite* sets. We will see how to deal with infinite sets of wffs of SL later on.

We will prove this claim using the method of conditional proof and mathematical induction. We will begin by assuming, of some arbitrary finite set of wffs Δ , that there is some truth-value assignment which makes all of the wffs in Δ true. From there, we will show that any tree Δ which has all the wffs in Δ at its root will have a consistent branch. We will show this by utilizing mathematical induction. Here is how the induction will proceed: we will take an arbitrary completed tree, Δ , with the wffs in Δ at its root, and we will use induction to show that, at every line of the tree, there is some branch with the following property: all the wffs which appear on that branch on or before that line are made true by some truth-value assignment. That's a mouthful. Let's see if we can make it a bit easier to say. To begin with, let's say that, if there is a truth-value assignment which makes all of the wffs appearing on a branch *at or before line n* true, then that branch is LINE- n CONSISTENT.

LINE-*n* BRANCH CONSISTENCY A branch of a tree is **LINE-*n* CONSISTENT** iff there is some truth-value assignment which makes all of the wffs appearing on that branch at or before line *n* true.

And let's say that, if a line, *n*, has a branch which is line-*n* consistent, then that *line* is consistent.

LINE CONSISTENCY A line, *n*, of a tree is consistent iff there is some branch at line *n* which is line-*n* consistent.

Then, what we will show with mathematical induction is that every line of the tree Δ is consistent. This tells us that, at every line, *n*, of the tree, there is some branch which is line-*n* consistent.

If, at every line, *n*, there is some branch which is line-*n* consistent, then there is some branch which is consistent (if the branch were not consistent, then there would be some line at which it became inconsistent; so, if we know that the branch is consistent at every line, then we know that the branch is consistent).¹

A short word on this terminology: an open branch may end on line 10, while another branch continues on to line 20 and eventually closes. We *don't* want to say that, in that case, there is no open branch at line 20, or that line 20 is not consistent. Rather, we will simply stipulate that any open branch extends to the end of the tree. Then, even if the last wff of the only open branch on a tree appears on line 10, if the tree extends to line 20, then there is an open branch at line 20; it's just that no new wffs have been added to it. And line 20 is still consistent.

Alright. That's the end of the broad outline of the proof. Now let's get into the nitty-gritty.

8.3 Proof that the Tree Method for *SL* is Sound

Proof. Assume, for the purposes of conditional proof, that there is some truth-value assignment which makes every wff in some arbitrary finite set Δ true.

Now, take an arbitrary tree, Δ , which has all of the wffs in Δ at its root. We will do mathematical induction on the lines of this tree. That is, our list is the following:

¹ Why not just say "there is some branch which is consistent at the *final* line of the tree"? Well, we haven't proven that every tree has a final line—and, indeed, if we want to carry over this proof technique to establish soundness for *PL* trees, it will be false that every tree has a final line. Nevertheless, it will still be true that, if, at every line *n*, there is some branch which is line-*n* consistent, then there is some branch which is consistent. (If this is confusing, don't worry about it for now; we'll prove it later on in the course).

- 0) The lines at the root of Δ on which the members of Δ appear.
- 1) The first line of Δ after the members of Δ .
- 2) The second line of Δ after the members of Δ .
- 3) The third line of Δ after the members of Δ .
- \vdots
- n The n th line of Δ after the members of Δ .

Our inductive property will be the following:

INDUCTIVE PROPERTY A line of a tree has the inductive property iff that line is consistent. (That is, iff, at that line there is some branch such that all the wffs on that branch appearing at or before the line are made true on some truth-value assignment.)

We will now show that every line of the tree Δ is consistent. That is: for every line of the tree, there is some branch at that line such that every wff which appears on that branch at or before that line is made true on some one truth-value assignment.

First, we complete the basis step by showing that the lines of Δ on which the wffs from Δ appear are consistent.

Basis Step. By assumption, the wffs in Δ are consistent. Every line at the root of Δ is on a branch such that the only wffs appearing at or before that line of the tree are lines on which the wffs from Δ appear. Therefore, for every line at the root of the tree, there will be some truth-value assignment which makes all the wffs appearing at or before that line true.

Next, we will complete the inductive step by showing that, for any k , if line k is consistent, then line $k + 1$ will be consistent as well.

Inductive Step. Assume, for the purposes of conditional proof, the **inductive hypothesis**.

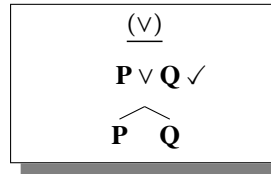
Inductive Hypothesis. For some arbitrary k , line k is consistent.

From the inductive hypothesis, there is at least one branch at line k such that every wff on that branch appearing at or before line k is made true by some truth-value assignment. Take one such branch—call it ' \mathbf{b} '—and take some truth-value assignment which makes all the wffs at or before line k on \mathbf{b} true. Call it ' \mathcal{A} '. If the last wff on \mathbf{b} appears on or before line k , then \mathbf{b} is consistent at line $k + 1$, and we are done. If, on the other hand,

some new wffs appear on \mathbf{b} at line $k + 1$, then \mathbf{b} must be extended in accordance with one of the tree rules. We will show that, no matter which rule is used, there will be some branch at line $k + 1$ such that the truth-value assignment \mathcal{A} makes every wff appearing at or before line $k + 1$ on that branch true. That is: no matter which rule is used, line $k + 1$ is consistent.

We will proceed by considering the possible cases.

Case 1: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for disjunction.



Then, since $\lceil \mathbf{P} \vee \mathbf{Q} \rceil$ appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \mathbf{P} \vee \mathbf{Q} \rceil$ true. By the definition of \vee , this could only be so if \mathcal{A} either makes $\lceil \mathbf{P} \rceil$ true or makes $\lceil \mathbf{Q} \rceil$ true.

\mathbf{P}	\mathbf{Q}	$\mathbf{P} \vee \mathbf{Q}$
T	T	T
T	F	T
F	T	T
F	F	F

- (a) Suppose that \mathcal{A} makes $\lceil \mathbf{P} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \mathbf{P} \rceil$. So, every wff appearing on the left-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.
- (b) Suppose, on the other hand, that \mathcal{A} makes $\lceil \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \mathbf{Q} \rceil$. So, every wff appearing on the right-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

So, either way, line $k + 1$ is consistent.

Case 2: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for conjunction.

($\&$)
$\mathbf{P \& Q} \checkmark$
\mathbf{P}
\mathbf{Q}

Then, since ' $\mathbf{P \& Q}$ ' appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes ' $\mathbf{P \& Q}$ ' true. By the definition of ' $\&$ ', this could only be so if \mathcal{A} both makes ' \mathbf{P} ' true and makes ' \mathbf{Q} ' true.

\mathbf{P}	\mathbf{Q}	$\mathbf{P \& Q}$
T	T	T
T	F	F
F	T	F
F	F	F

And both ' \mathbf{P} ' and ' \mathbf{Q} ' appear on \mathbf{b} at line $k + 1$. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on \mathbf{b} . So, every wff appearing on \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

Case 3: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for the conditional.

(\supset)
$\mathbf{P \supset Q} \checkmark$
^
$\sim\mathbf{P} \quad \mathbf{Q}$

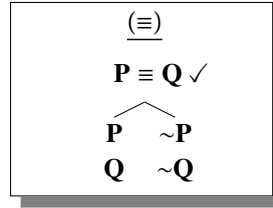
Then, since ' $\mathbf{P \supset Q}$ ' appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes ' $\mathbf{P \supset Q}$ ' true. By the definition of ' \supset ', this could only be so if \mathcal{A} either makes ' \mathbf{P} ' false or makes ' \mathbf{Q} ' true.

\mathbf{P}	\mathbf{Q}	$\mathbf{P \supset Q}$
T	T	T
T	F	F
F	T	T
F	F	T

- (a) Suppose that \mathcal{A} makes $\lceil \mathbf{P} \rceil$ false. Then, by the definition of $\lceil \sim \rceil$, \mathcal{A} makes $\lceil \sim \mathbf{P} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \sim \mathbf{P} \rceil$. So, every wff appearing on the left-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.
- (b) Suppose, on the other hand, that \mathcal{A} makes $\lceil \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \mathbf{Q} \rceil$. So, every wff appearing on the right-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

So, either way, line $k + 1$ is consistent.

Case 4: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for the biconditional.



Then, since $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \mathbf{P} \equiv \mathbf{Q} \rceil$ true. By the definition of $\lceil \equiv \rceil$, this could only be so if \mathcal{A} either makes both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ true or makes both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ false.

\mathbf{P}	\mathbf{Q}	$\mathbf{P} \equiv \mathbf{Q}$
T	T	T
T	F	F
F	T	F
F	F	T

- (a) Suppose that \mathcal{A} makes both $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$. So, every wff appearing on the left-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.
- (b) Suppose, on the other hand, that \mathcal{A} makes $\lceil \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ false. Then, by the definition of $\lceil \sim \rceil$, \mathcal{A} makes $\lceil \sim \mathbf{P} \rceil$ and $\lceil \sim \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \sim \mathbf{P} \rceil$ and $\lceil \sim \mathbf{Q} \rceil$. So, every wff appearing on the right-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

So, either way, line $k + 1$ is consistent.

Case 5: Suppose that the branch **b** is extended at line $k + 1$ by an application of the rule for negated negations.

$\frac{(\sim\sim)}{\sim\sim\mathbf{P} \checkmark}$ $\quad \mid$ $\quad \mathbf{P}$
--

Then, since $\lceil \sim\sim\mathbf{P} \rceil$ appears on **b** at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \sim\sim\mathbf{P} \rceil$ true. By the definition of ' \sim ', this could only be so if \mathcal{A} makes $\lceil \mathbf{P} \rceil$ true.

P	$\sim\mathbf{P}$	$\sim\sim\mathbf{P}$
<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>

$\lceil \mathbf{P} \rceil$ appears on **b** at line $k + 1$. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on **b**. So, every wff appearing on **b** at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

Case 6: Suppose that the branch **b** is extended at line $k + 1$ by an application of the rule for negated disjunctions.

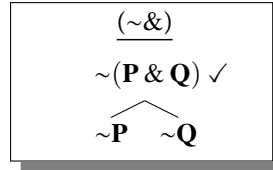
$\frac{(\sim\vee)}{\sim(\mathbf{P}\vee\mathbf{Q}) \checkmark}$ $\quad \mid$ $\quad \sim\mathbf{P}$ $\quad \sim\mathbf{Q}$

Then, since $\lceil \sim(\mathbf{P}\vee\mathbf{Q}) \rceil$ appears on **b** at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \sim(\mathbf{P}\vee\mathbf{Q}) \rceil$ true. By the definition of ' \sim ' and ' \vee ', this could only be so if \mathcal{A} makes $\lceil \sim\mathbf{P} \rceil$ and $\lceil \sim\mathbf{Q} \rceil$ true.

P	Q	$\sim(\mathbf{P}\vee\mathbf{Q})$	$\sim\mathbf{P}$	$\sim\mathbf{Q}$
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>

$\lceil \sim \mathbf{P} \rceil$ and $\lceil \sim \mathbf{Q} \rceil$ appear on \mathbf{b} at line $k + 1$. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on \mathbf{b} . So, every wff appearing on \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

Case 7: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for negated conjunctions.



Then, since $\lceil \sim(\mathbf{P} \& \mathbf{Q}) \rceil$ appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \sim(\mathbf{P} \& \mathbf{Q}) \rceil$ true. By the definition of ' \sim ' and ' $\&$ ', this could only be so if \mathcal{A} either makes $\lceil \sim \mathbf{P} \rceil$ true or makes $\lceil \sim \mathbf{Q} \rceil$ true.

\mathbf{P}	\mathbf{Q}	$\sim(\mathbf{P} \& \mathbf{Q})$	$\sim \mathbf{P}$	$\sim \mathbf{Q}$
T	T	F	F	F
T	F	T	F	T
F	T	T	T	F
F	F	T	T	T

- (a) Suppose that \mathcal{A} makes $\lceil \sim \mathbf{P} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \sim \mathbf{P} \rceil$. So, every wff appearing on the left-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.
- (b) Suppose, on the other hand, that \mathcal{A} makes $\lceil \sim \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \sim \mathbf{Q} \rceil$. So, every wff appearing on the right-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

So, either way, line $k + 1$ is consistent.

Case 8: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for negated conditionals.

$(\sim \supset)$
$\sim(\mathbf{P} \supset \mathbf{Q}) \checkmark$
\mathbf{P}
$\sim\mathbf{Q}$

Then, since $\lceil \sim(\mathbf{P} \supset \mathbf{Q}) \rceil$ appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \sim(\mathbf{P} \supset \mathbf{Q}) \rceil$ true. By the definition of $\lceil \sim \rceil$ and $\lceil \supset \rceil$, this could only be so if \mathcal{A} makes both $\lceil \mathbf{P} \rceil$ and $\lceil \sim\mathbf{Q} \rceil$ true.

\mathbf{P}	\mathbf{Q}	$\sim(\mathbf{P} \supset \mathbf{Q})$	$\sim\mathbf{Q}$
T	T	F	F
T	F	T	T
F	T	F	F
F	F	F	T

$\lceil \mathbf{P} \rceil$ and $\lceil \sim\mathbf{Q} \rceil$ appear on \mathbf{b} at line $k + 1$. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on \mathbf{b} . So, every wff appearing on \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

Case 9: Suppose that the branch \mathbf{b} is extended at line $k + 1$ by an application of the rule for negated biconditionals.

$(\sim \equiv)$
$\sim(\mathbf{P} \equiv \mathbf{Q}) \checkmark$
/ \
$\mathbf{P} \quad \sim\mathbf{P}$
$\sim\mathbf{Q} \quad \mathbf{Q}$

Then, since $\lceil \sim(\mathbf{P} \equiv \mathbf{Q}) \rceil$ appears on \mathbf{b} at or before line k , by the inductive hypothesis, the truth-value assignment \mathcal{A} makes $\lceil \sim(\mathbf{P} \equiv \mathbf{Q}) \rceil$ true. By the definition of $\lceil \sim \rceil$ and $\lceil \equiv \rceil$, this could only be so if \mathcal{A} either makes $\lceil \mathbf{P} \rceil$ true and $\lceil \mathbf{Q} \rceil$ false or makes $\lceil \mathbf{P} \rceil$ false and $\lceil \mathbf{Q} \rceil$ true.

\mathbf{P}	\mathbf{Q}	$\sim(\mathbf{P} \equiv \mathbf{Q})$
T	T	F
T	F	T
F	T	T
F	F	F

- (a) Suppose that \mathcal{A} makes $\lceil \mathbf{P} \rceil$ true and $\lceil \mathbf{Q} \rceil$ false. By the definition of $\lceil \sim \rceil$, this means that \mathcal{A} makes both $\lceil \mathbf{P} \rceil$ and $\lceil \sim \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \mathbf{P} \rceil$ and $\lceil \sim \mathbf{Q} \rceil$. So, every wff appearing on the left-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.
- (b) Suppose, on the other hand, that \mathcal{A} makes $\lceil \mathbf{P} \rceil$ false and $\lceil \mathbf{Q} \rceil$ true. By the definition of $\lceil \sim \rceil$, this means that \mathcal{A} makes both $\lceil \sim \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$ true. We already know, by the inductive hypothesis, that \mathcal{A} makes true every earlier wff on the same branch as $\lceil \sim \mathbf{P} \rceil$ and $\lceil \mathbf{Q} \rceil$. So, every wff appearing on the right-hand-side extension of \mathbf{b} at or before line $k + 1$ is made true by \mathcal{A} . So line $k + 1$ is consistent.

So, either way, line $k + 1$ is consistent.

We have now considered all the possible cases and seen that, in each case, line $k + 1$ is consistent. So, no matter which is the case, line $k + 1$ will be consistent.

Thus, from the inductive hypothesis that line k is consistent, we have shown that line $k + 1$ is consistent. Since k was arbitrary, we may conclude that, *for any* k , if line k is consistent, then line $k + 1$ is consistent.

We have completed the basis step and the inductive step. So we can conclude that every line of the completed tree Δ which has the wffs in Δ at its root is consistent. That is: for every line of the tree, there is some branch at that line such that every wff appearing on that branch at or before that line is made true on some one truth-value assignment.

So, there is some branch of the completed tree such that every wff appearing on that branch is made true by some one truth-value assignment. (If this step seems difficult to follow, don't worry—we will prove it more rigorously later on.)

But Δ and Δ were arbitrary. Thus, we can conclude that the same holds for any finite set Δ and any completed tree Δ with the wffs from Δ at its root. Thus, we have completed our proof of the claim:

Claim. *For any finite set of wffs of SL , Δ , if there is some truth-value assignment which makes every member of Δ true, then any tree \mathcal{T}_Δ which has the members of Δ at its root will have a consistent branch.*

Since the only way for a branch to close is for there to be wffs of the form $\lceil \mathbf{P} \rceil$ and $\lceil \sim \mathbf{P} \rceil$ on the same branch, and since there is no truth-value assignment which makes both

$\lceil \mathbf{P} \rceil$ and $\lceil \sim \mathbf{P} \rceil$ true, if there is a branch such that there is some truth-value assignment which makes every wff on that branch true, then that branch will not close. So, we know that the tree Δ has some branch which remains open. So Δ remains open.

So we may conclude that, *for any* finite set of wffs of *SL* Δ , and *any* *SL* tree Δ which has all and only the wffs in Δ at its root, if Δ is *SL*-consistent, then Δ remains open.

We've now proven that, in general, if a finite set of wffs of *SL* Δ is *SL*-consistent, then any *SL* tree with all and only the wffs in Δ at its root will remain open.

So, in particular, if Γ is a finite set of the premises of an *SL* argument and $\lceil \mathbf{P} \rceil$ is the conclusion of that *SL* argument, then, if $\Gamma \cup \{\sim \mathbf{P}\}$ is *SL*-consistent, then the tree with all and only the wffs in $\Gamma \cup \{\sim \mathbf{P}\}$ at its root will remain open.

if $\Gamma \cup \{\sim \mathbf{P}\}$ is *SL*-consistent, then $\Gamma \not\vdash_{SL} \mathbf{P}$

Taking the contrapositive of this conditional, we get

if $\Gamma \vdash_{SL} \mathbf{P}$, then $\Gamma \cup \{\sim \mathbf{P}\}$ is *SL*-inconsistent

And, since we know that $\Gamma \cup \{\sim \mathbf{P}\}$ is *SL*-inconsistent iff $\Gamma \models_{SL} \mathbf{P}$, this means that

if $\Gamma \vdash_{SL} \mathbf{P}$, then $\Gamma \models_{SL} \mathbf{P}$

That is: the *SL* tree method is sound (at least, if the premise set is finite).

Chapter 9

Completeness of the Tree Method for Sentence Logic

Today we wish to show that the tree method for *SL* is *complete*. Recall, the tree method is complete iff, whenever an argument is *SL*-valid, the tree method *tells us* that it is valid.

COMPLETENESS The tree method for *SL* is complete iff, for any set of wffs of *SL* Γ and any wff of *SL* \mathbf{P} :

$$\text{if } \Gamma \models_{SL} \mathbf{P}, \text{ then } \Gamma \vdash_{SL} \mathbf{P}$$

This is, recall, equivalent to saying that we can trust everything that the tree method tells us about *invalidity*. That's because, in general, $\mathbf{P} \supset \mathbf{Q}$ is equivalent to $\sim \mathbf{Q} \supset \sim \mathbf{P}$ —known as the *contrapositive* of $\mathbf{P} \supset \mathbf{Q}$. Therefore, we can re-write the property of completeness as follows:

$$\text{if } \Gamma \not\vdash_{SL} \mathbf{P}, \text{ then } \Gamma \not\models_{SL} \mathbf{P} \quad (\text{COMPLETENESS})$$

That is: if the tree method says that an argument is invalid, then it is invalid.

Recall: the argument from the premises in Γ to the conclusion \mathbf{P} is *SL*-invalid iff there is some truth-value assignment which makes all the wffs in Γ true and which makes \mathbf{P} false. And this is so iff there is some truth-value assignment which makes all the wffs in Γ true and which makes $\sim \mathbf{P}$ true. And this is true iff there is some truth-value assignment which makes all the wffs in $\Gamma \cup \{\sim \mathbf{P}\}$ true. Therefore, $\Gamma \not\vdash_{SL} \mathbf{P}$ iff $\Gamma \cup \{\sim \mathbf{P}\}$ is *SL*-consistent. So we can re-write completeness as follows:

$$\text{if } \Gamma \not\vdash_{SL} \mathbf{P}, \text{ then } \Gamma \cup \{\sim \mathbf{P}\} \text{ is } SL\text{-consistent} \quad (\text{COMPLETENESS})$$

This is the form in which we will prove that the tree method for SL is complete. We will show that this claim holds for any finite set of premises Γ and any conclusion $\lceil \mathbf{P} \rceil$. We will show this by showing something more informative. We will show that *for any* finite set of wffs Δ , if the tree beginning with all and only the wffs in Δ has an open branch, then Δ is SL -consistent. That is, we will prove the following claim

Claim 9. *For any finite set of wffs Δ , and any tree Δ which begins with all and only the wffs in Δ at its root, if Δ has an open branch, then there is some truth-value assignment which makes all the wffs in Δ true.*

Note that, in the claim above, we have restricted our attention to *finite* sets of wffs. We will talk about how to relax this restriction later on in the course.

9.1 The Proof in Broad Outline

Here's how our proof of completeness is going to proceed. The claim we wish to establish is a universally quantified conditional. We're going to prove it by using conditional proof. That is, we will assume, for some *arbitrary* finite set of wffs Δ and some *arbitrary* SL tree Δ with all and only the wffs in Δ at its root, that Δ has an open branch. Call that open branch ' \mathbf{b} '. We will use this open branch to construct a truth-value assignment $\mathcal{A}_{\mathbf{b}}$ which makes all of the wffs appearing on \mathbf{b} true. Since the wffs in Δ are at the root of the tree, they appear on every branch. Therefore, $\mathcal{A}_{\mathbf{b}}$ makes all the wffs in Δ true. Therefore, there is some truth-value assignment which makes all the wffs in Δ true. Therefore, Δ is SL -consistent.

Thus, we will have proven, from our assumption that a tree Δ with all and only the wffs in Δ at its root has an open branch, that Δ is SL -consistent. Since Δ and Δ were arbitrary, we can conclude that, *for any finite set Δ and any tree Δ with all and only the wffs in Δ at its root, if Δ has an open branch, then Δ is SL -consistent.*

I said that we would show that, if there's an open branch, ' \mathbf{b} ', on Δ , then there's some truth-value assignment $\mathcal{A}_{\mathbf{b}}$ which makes all the wffs appearing on \mathbf{b} true. How will we show this? We will use mathematical induction. First, we will construct the truth-value assignment $\mathcal{A}_{\mathbf{b}}$ as follows: if the negation of a sentence letter appears on \mathbf{b} , then that statement letter will be given the truth-value 'false' by $\mathcal{A}_{\mathbf{b}}$. Every other statement letter of SL will be given the truth-value 'true' by $\mathcal{A}_{\mathbf{b}}$. We will then show that every wff on \mathbf{b} which is a statement letter is true on $\mathcal{A}_{\mathbf{b}}$ (this is the basis step). And we will show that, for any $k \geq 0$, if every wff on \mathbf{b} with k or fewer symbols from the vocabulary of SL is true on $\mathcal{A}_{\mathbf{b}}$, then every wff on \mathbf{b} with $k + 1$ symbols from the vocabulary of SL is true on $\mathcal{A}_{\mathbf{b}}$ is true too (this is the inductive step).

From the basis step and the inductive step, we can conclude that every wff on \mathbf{b} is true

on $\mathcal{A}_{\mathbf{b}}$. Since all the wffs in Δ are on \mathbf{b} , all the wffs in Δ are true on $\mathcal{A}_{\mathbf{b}}$. So, Δ is SL -consistent.

9.2 Proof that the Tree Method for SL is Complete

Proof. Assume, for the purposes of conditional proof, that there is some complete tree Δ which has all and only the wffs in Δ at its root and which has at least one open branch. Take the left-most open branch of Δ . Call that branch ' \mathbf{b} '. From \mathbf{b} we may build a truth-value assignment $\mathcal{A}_{\mathbf{b}}$ as follows:

1. For any statement letter ' \mathbf{A} ', if ' $\sim\mathbf{A}$ ' appears on \mathbf{b} , then let ' \mathbf{A} ' receive the truth-value 'false' on $\mathcal{A}_{\mathbf{b}}$.
2. For any statement letter ' \mathbf{A} ', if ' $\sim\mathbf{A}$ ' does not appear on \mathbf{b} , then let ' \mathbf{A} ' receive the truth-value 'true' on $\mathcal{A}_{\mathbf{b}}$.

We will be doing mathematical induction on the following list:

- set 0) All statement letters on \mathbf{b} .
- set 1) All wffs on \mathbf{b} with 2 symbols from the vocabulary of SL .
- set 2) All wffs on \mathbf{b} with 3 symbols from the vocabulary of SL .
- ⋮
- set n) All wffs on \mathbf{b} with n symbols from the vocabulary of SL .
- ⋮

If a wff of SL has n symbols from the vocabulary of SL , then we will say that that wff has *length* n .

And we will be using the following inductive property:

INDUCTIVE PROPERTY A set from our list has the inductive property iff that set contains only wffs which are true on the truth-value assignment $\mathcal{A}_{\mathbf{b}}$.

First, we must complete the basis step by showing that every wff in set 0 is true on $\mathcal{A}_{\mathbf{b}}$.

Basis Step. For any statement letter $\ulcorner A \urcorner$ appearing on \mathbf{b} , $\mathcal{A}_{\mathbf{b}}$ would only give $\ulcorner A \urcorner$ the truth-value ‘false’ if $\ulcorner \sim A \urcorner$ appeared on \mathbf{b} . If both $\ulcorner A \urcorner$ and $\ulcorner \sim A \urcorner$ appeared on \mathbf{b} , then \mathbf{b} would be a closed branch. But we know that \mathbf{b} is an open branch. So, for any statement letter $\ulcorner A \urcorner$ appearing on \mathbf{b} , we know that $\ulcorner \sim A \urcorner$ does not appear on \mathbf{b} . Therefore, we know that $\mathcal{A}_{\mathbf{b}}$ gives any statement letter appearing on \mathbf{b} the truth-value ‘true’. Therefore, every wff in set 0 is true on $\mathcal{A}_{\mathbf{b}}$.

Now, we complete the inductive step by showing that, for any k , if all wffs on \mathbf{b} of length k or lower are true on $\mathcal{A}_{\mathbf{b}}$, then all wffs on \mathbf{b} with $k + 1$ symbols from the vocabulary of SL will also be true on $\mathcal{A}_{\mathbf{b}}$.

Inductive Step. Assume, for the purposes of conditional proof, the **inductive hypothesis**.

Inductive Hypothesis. For some arbitrary k , all wffs on \mathbf{b} of length k or lower are true on $\mathcal{A}_{\mathbf{b}}$.

There may very well not be any wffs of length $k + 1$ on \mathbf{b} . If so, then all of the none of them are trivially true on $\mathcal{A}_{\mathbf{b}}$. If, on the other hand, there are some wffs of length $k + 1$ on \mathbf{b} , then any such wff, $\ulcorner P \urcorner$, will have one of the following forms:

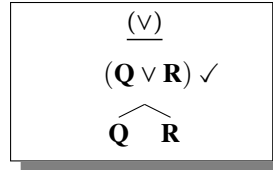
1. $\ulcorner \sim A \urcorner$, for some statement letter $\ulcorner A \urcorner$.
2. $\ulcorner (Q \vee R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 2$.
3. $\ulcorner (Q \& R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 2$.
4. $\ulcorner (Q \supset R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 2$.
5. $\ulcorner (Q \equiv R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 2$.
6. $\ulcorner \sim \sim Q \urcorner$, for some $\ulcorner Q \urcorner$ of length less than or equal to $k - 1$.
7. $\ulcorner \sim(Q \vee R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 3$.
8. $\ulcorner \sim(Q \& R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 3$.
9. $\ulcorner \sim(Q \supset R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 3$.
10. $\ulcorner \sim(Q \equiv R) \urcorner$, for some $\ulcorner Q \urcorner$, $\ulcorner R \urcorner$ of length less than or equal to $k - 3$.

We will show that, in each case, $\ulcorner P \urcorner$ will be true on $\mathcal{A}_{\mathbf{b}}$.

Case 1: Suppose that $\ulcorner P \urcorner$ is of the form $\ulcorner \sim A \urcorner$, for some statement letter $\ulcorner A \urcorner$. Then, by the definition of $\mathcal{A}_{\mathbf{b}}$, $\ulcorner A \urcorner$ is given the value ‘false’ by $\mathcal{A}_{\mathbf{b}}$. By the definition of ‘ \sim ’, $\ulcorner \sim A \urcorner$ is given the value ‘true’ by $\mathcal{A}_{\mathbf{b}}$.

So $\ulcorner P \urcorner$ is true on $\mathcal{A}_{\mathbf{b}}$.

Case 2: Suppose that $\ulcorner \mathbf{P} \urcorner$ is of the form $\ulcorner (\mathbf{Q} \vee \mathbf{R}) \urcorner$, for some $\ulcorner \mathbf{Q} \urcorner$, $\ulcorner \mathbf{R} \urcorner$ of length less than or equal to $k - 2$. Then, since $\ulcorner (\mathbf{Q} \vee \mathbf{R}) \urcorner$ appears on an open branch, \mathbf{b} , $\ulcorner (\mathbf{Q} \vee \mathbf{R}) \urcorner$ must have had the rule (\vee) applied to it (else the tree would not be complete).

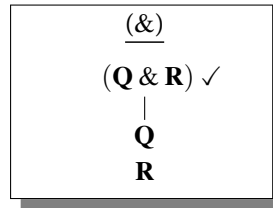


Since $\ulcorner (\mathbf{Q} \vee \mathbf{R}) \urcorner$ is on \mathbf{b} , either $\ulcorner \mathbf{Q} \urcorner$ is on \mathbf{b} or $\ulcorner \mathbf{R} \urcorner$ is on \mathbf{b} .

- (a) If $\ulcorner \mathbf{Q} \urcorner$ is on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner \mathbf{Q} \urcorner$ true. And, by the definition of ' \vee ', if $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner \mathbf{Q} \urcorner$ true, then $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner (\mathbf{Q} \vee \mathbf{R}) \urcorner$ true.
- (b) If $\ulcorner \mathbf{R} \urcorner$ is on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner \mathbf{R} \urcorner$ true. And, by the definition of ' \vee ', if $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner \mathbf{R} \urcorner$ true, then $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner (\mathbf{Q} \vee \mathbf{R}) \urcorner$ true.

So, either way, $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner \mathbf{P} \urcorner$ true.

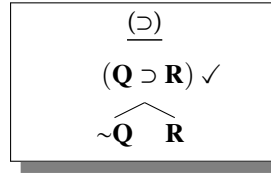
Case 3: Suppose that $\ulcorner \mathbf{P} \urcorner$ is of the form $\ulcorner (\mathbf{Q} \& \mathbf{R}) \urcorner$, for some $\ulcorner \mathbf{Q} \urcorner$, $\ulcorner \mathbf{R} \urcorner$ of length less than or equal to $k - 2$. Then, since $\ulcorner (\mathbf{Q} \& \mathbf{R}) \urcorner$ appears on an open branch, \mathbf{b} , $\ulcorner (\mathbf{Q} \& \mathbf{R}) \urcorner$ must have had the rule $(\&)$ applied to it (else the tree would not be complete).



Since $\ulcorner (\mathbf{Q} \& \mathbf{R}) \urcorner$ is on \mathbf{b} , both $\ulcorner \mathbf{Q} \urcorner$ and $\ulcorner \mathbf{R} \urcorner$ are on \mathbf{b} . By the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes both $\ulcorner \mathbf{Q} \urcorner$ and $\ulcorner \mathbf{R} \urcorner$ true. By the definition of ' $\&$ ', if $\mathcal{A}_{\mathbf{b}}$ makes both $\ulcorner \mathbf{Q} \urcorner$ and $\ulcorner \mathbf{R} \urcorner$ true, then $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner (\mathbf{Q} \& \mathbf{R}) \urcorner$ true.

So, $\mathcal{A}_{\mathbf{b}}$ makes $\ulcorner \mathbf{P} \urcorner$ true.

Case 4: Suppose that $\ulcorner \mathbf{P} \urcorner$ is of the form $\ulcorner (\mathbf{Q} \supset \mathbf{R}) \urcorner$, for some $\ulcorner \mathbf{Q} \urcorner$, $\ulcorner \mathbf{R} \urcorner$ of length less than or equal to $k - 2$. Then, since $\ulcorner (\mathbf{Q} \supset \mathbf{R}) \urcorner$ appears on an open branch, \mathbf{b} , $\ulcorner (\mathbf{Q} \supset \mathbf{R}) \urcorner$ must have had the rule (\supset) applied to it (else the tree would not be complete).

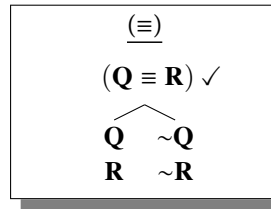


Since ' $\mathbf{Q} \supset \mathbf{R}$ ' is on \mathbf{b} , either ' $\sim \mathbf{Q}$ ' is on \mathbf{b} or ' \mathbf{R} ' is on \mathbf{b} .

- (a) If ' $\sim \mathbf{Q}$ ' is on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes ' $\sim \mathbf{Q}$ ' true. By the definition of ' \sim ', $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{Q} ' false. By the definition of ' \supset ', if $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{Q} ' false, then $\mathcal{A}_{\mathbf{b}}$ makes ' $\mathbf{Q} \supset \mathbf{R}$ ' true.
- (b) If ' \mathbf{R} ' is on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{R} ' true. And, by the definition of ' \supset ', if $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{R} ' true, then $\mathcal{A}_{\mathbf{b}}$ makes ' $\mathbf{Q} \supset \mathbf{R}$ ' true.

So, either way, $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{P} ' true.

Case 5: Suppose that ' \mathbf{P} ' is of the form ' $\mathbf{Q} \equiv \mathbf{R}$ ', for some ' \mathbf{Q} ', ' \mathbf{R} ' of length less than or equal to $k - 2$. Then, since ' $\mathbf{Q} \equiv \mathbf{R}$ ' appears on an open branch, \mathbf{b} , ' $\mathbf{Q} \equiv \mathbf{R}$ ' must have had the rule (\equiv) applied to it (else the tree would not be complete).

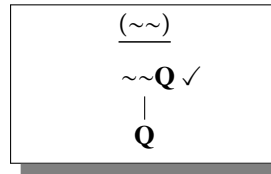


Since ' $\mathbf{Q} \equiv \mathbf{R}$ ' is on \mathbf{b} , either both ' \mathbf{Q} ' and ' \mathbf{R} ' are on \mathbf{b} or both ' $\sim \mathbf{Q}$ ' and ' $\sim \mathbf{R}$ ' are on \mathbf{b} .

- (a) If both ' \mathbf{Q} ' and ' \mathbf{R} ' are on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes both ' \mathbf{Q} ' and ' \mathbf{R} ' true. By the definition of ' \equiv ', if $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{Q} ' and ' \mathbf{R} ' true, then $\mathcal{A}_{\mathbf{b}}$ makes ' $\mathbf{Q} \equiv \mathbf{R}$ ' true.
- (b) If both ' $\sim \mathbf{Q}$ ' and ' $\sim \mathbf{R}$ ' are on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes both ' $\sim \mathbf{Q}$ ' and ' $\sim \mathbf{R}$ ' true. By the definition of ' \sim ', $\mathcal{A}_{\mathbf{b}}$ makes both ' \mathbf{Q} ' and ' \mathbf{R} ' false. By the definition of ' \equiv ', if $\mathcal{A}_{\mathbf{b}}$ makes both ' \mathbf{Q} ' and ' \mathbf{R} ' true, then $\mathcal{A}_{\mathbf{b}}$ makes ' $\mathbf{Q} \equiv \mathbf{R}$ ' true.

So, either way, $\mathcal{A}_{\mathbf{b}}$ makes ' \mathbf{P} ' true.

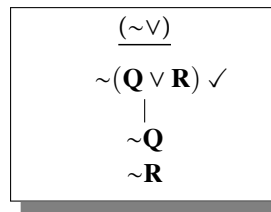
Case 6: Suppose that ' \mathbf{P} ' is of the form ' $\sim \sim \mathbf{Q}$ ', for some ' \mathbf{Q} ' of length less than or equal to $k - 1$. Then, since ' $\sim \sim \mathbf{Q}$ ' appears on an open branch, ' $\sim \sim \mathbf{Q}$ ' must have had the rule ($\sim \sim$) applied to it (else the tree would not be complete).



Since $\lceil \sim\sim\mathbf{Q} \rceil$ is on \mathbf{b} , $\lceil \mathbf{Q} \rceil$ must be on \mathbf{b} . By the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{Q} \rceil$ true. By the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ must make $\lceil \sim\sim\mathbf{Q} \rceil$ true as well.

So, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{P} \rceil$ true.

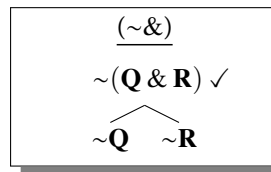
Case 7: Suppose that $\lceil \mathbf{P} \rceil$ is of the form $\lceil \sim(\mathbf{Q} \vee \mathbf{R}) \rceil$, for some $\lceil \mathbf{Q} \rceil$, $\lceil \mathbf{R} \rceil$ of length less than or equal to $k - 3$. Then, since $\lceil \sim(\mathbf{Q} \vee \mathbf{R}) \rceil$ appears on an open branch, \mathbf{b} , $\lceil \sim(\mathbf{Q} \vee \mathbf{R}) \rceil$ must have had the rule $(\sim\vee)$ applied to it (else the tree would not be complete).



Since $\lceil \sim(\mathbf{Q} \vee \mathbf{R}) \rceil$ is on \mathbf{b} , both $\lceil \sim\mathbf{Q} \rceil$ and $\lceil \sim\mathbf{R} \rceil$ must be on \mathbf{b} too. By the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes both $\lceil \sim\mathbf{Q} \rceil$ and $\lceil \sim\mathbf{R} \rceil$ true. By the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes both $\lceil \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ false. Thus, by the definition of $\lceil \vee \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil (\mathbf{Q} \vee \mathbf{R}) \rceil$ false. Thus, by the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \sim(\mathbf{Q} \vee \mathbf{R}) \rceil$ true.

So, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{P} \rceil$ true.

Case 8: Suppose that $\lceil \mathbf{P} \rceil$ is of the form $\lceil \sim(\mathbf{Q} \& \mathbf{R}) \rceil$, for some $\lceil \mathbf{Q} \rceil$, $\lceil \mathbf{R} \rceil$ of length less than or equal to $k - 3$. Then, since $\lceil \sim(\mathbf{Q} \& \mathbf{R}) \rceil$ appears on an open branch, \mathbf{b} , $\lceil \sim(\mathbf{Q} \& \mathbf{R}) \rceil$ must have had the rule $(\sim\&)$ applied to it (else the tree would not be complete).

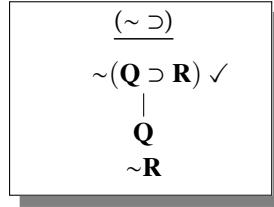


Since $\lceil \sim(\mathbf{Q} \& \mathbf{R}) \rceil$ is on \mathbf{b} , either $\lceil \sim\mathbf{Q} \rceil$ is on \mathbf{b} or $\lceil \sim\mathbf{R} \rceil$ is on \mathbf{b} .

- (a) If $\lceil \sim Q \rceil$ is on **b**, then, by the inductive hypothesis, \mathcal{A}_b makes $\lceil \sim Q \rceil$ true. By the definition of ' \sim ', if \mathcal{A}_b makes $\lceil \sim Q \rceil$ true, then \mathcal{A}_b makes $\lceil Q \rceil$ false. By the definition of '&', if \mathcal{A}_b makes $\lceil Q \rceil$ false, then \mathcal{A}_b makes $\lceil (Q \& R) \rceil$ false. By the definition of ' \sim ', if \mathcal{A}_b makes $\lceil (Q \& R) \rceil$ false, then \mathcal{A}_b makes $\lceil \sim(Q \& R) \rceil$ true.
- (b) If $\lceil \sim R \rceil$ is on **b**, then, by the inductive hypothesis, \mathcal{A}_b makes $\lceil \sim R \rceil$ true. By the definition of ' \sim ', \mathcal{A}_b makes $\lceil R \rceil$ false. By the definition of '&', if \mathcal{A}_b makes $\lceil R \rceil$ false, then \mathcal{A}_b makes $\lceil (Q \& R) \rceil$ false. By the definition of ' \sim ', \mathcal{A}_b makes $\lceil \sim(Q \& R) \rceil$ true.

So, either way, \mathcal{A}_b makes $\lceil P \rceil$ true.

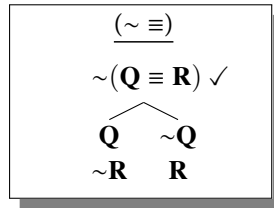
Case 9: Suppose that $\lceil P \rceil$ is of the form $\lceil \sim(Q \supset R) \rceil$, for some $\lceil Q \rceil$, $\lceil R \rceil$ of length less than or equal to $k - 3$. Then, since $\lceil \sim(Q \supset R) \rceil$ appears on an open branch, **b**, $\lceil \sim(Q \supset R) \rceil$ must have had the rule $(\sim \supset)$ applied to it (else the tree would not be complete).



Since $\lceil \sim(Q \supset R) \rceil$ is on **b**, both $\lceil Q \rceil$ and $\lceil \sim R \rceil$ must be on **b** too. By the inductive hypothesis, \mathcal{A}_b makes both $\lceil Q \rceil$ and $\lceil \sim R \rceil$ true. By the definition of ' \sim ', \mathcal{A}_b makes $\lceil R \rceil$ false. Thus, by the definition of ' \supset ', \mathcal{A}_b makes $\lceil (Q \supset R) \rceil$ false. Thus, by the definition of ' \sim ', \mathcal{A}_b makes $\lceil \sim(Q \supset R) \rceil$ true.

So, \mathcal{A}_b makes $\lceil P \rceil$ true.

Case 10: Suppose that $\lceil P \rceil$ is of the form $\lceil \sim(Q \equiv R) \rceil$, for some $\lceil Q \rceil$, $\lceil R \rceil$ of length less than or equal to $k - 3$. Then, since $\lceil \sim(Q \equiv R) \rceil$ appears on an open branch, **b**, $\lceil \sim(Q \equiv R) \rceil$ must have had the rule $(\sim \equiv)$ applied to it (else the tree would not be complete).



Since $\lceil \sim(Q \equiv R) \rceil$ is on **b**, either both $\lceil Q \rceil$ and $\lceil \sim R \rceil$ are on **b** or both $\lceil \sim Q \rceil$ and $\lceil R \rceil$ are on **b**.

- (a) If both $\lceil \mathbf{Q} \rceil$ and $\lceil \sim \mathbf{R} \rceil$ are on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes both $\lceil \mathbf{Q} \rceil$ and $\lceil \sim \mathbf{R} \rceil$ true. By the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{R} \rceil$ false. By the definition of $\lceil \equiv \rceil$, if $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{Q} \rceil$ true and $\lceil \mathbf{R} \rceil$ false, then $\mathcal{A}_{\mathbf{b}}$ makes $\lceil (\mathbf{Q} \equiv \mathbf{R}) \rceil$ false. By the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \sim (\mathbf{Q} \equiv \mathbf{R}) \rceil$ true.
- (b) If both $\lceil \sim \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ are on \mathbf{b} , then, by the inductive hypothesis, $\mathcal{A}_{\mathbf{b}}$ makes both $\lceil \sim \mathbf{Q} \rceil$ and $\lceil \mathbf{R} \rceil$ true. By the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{Q} \rceil$ false. By the definition of $\lceil \equiv \rceil$, if $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{Q} \rceil$ false and $\lceil \mathbf{R} \rceil$ true, then $\mathcal{A}_{\mathbf{b}}$ makes $\lceil (\mathbf{Q} \equiv \mathbf{R}) \rceil$ false. By the definition of $\lceil \sim \rceil$, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \sim (\mathbf{Q} \equiv \mathbf{R}) \rceil$ true.

So, either way, $\mathcal{A}_{\mathbf{b}}$ makes $\lceil \mathbf{P} \rceil$ true.

Therefore, for any wff $\lceil \mathbf{P} \rceil$ on \mathbf{b} of length $k + 1$, no matter its form, it follows from the inductive hypothesis (that all wffs on \mathbf{b} of length less than or equal to k are true on $\mathcal{A}_{\mathbf{b}}$) that $\lceil \mathbf{P} \rceil$ is true on $\mathcal{A}_{\mathbf{b}}$.

Since k was arbitrary, we may conclude that, for any k , if all wffs on \mathbf{b} of length less than or equal to k are true on $\mathcal{A}_{\mathbf{b}}$, then all wffs on \mathbf{b} of length $k + 1$ are true on $\mathcal{A}_{\mathbf{b}}$, too.

We have now completed the basis step and the inductive step. We have therefore shown, from our assumption that a tree Δ with all and only the wffs in Δ at its root has an open branch, \mathbf{b} , that there is some truth-value assignment which makes all the wffs on that branch true. However, Δ and Δ were arbitrary. We may therefore conclude:

Claim 1. *For any finite set of wffs Δ , and any tree Δ which begins with all and only the wffs in Δ at its root, if Δ has an open branch, then there is some truth-value assignment which makes all the wffs in Δ true.*

Claim 1 tells us that, if a tree beginning with the wffs in Δ has an open branch, then the wffs appearing on that branch are SL -consistent. In particular, since all the wffs in Δ are on every branch of Δ , **Claim 1** tells us that Δ is SL -consistent.

If we let $\Delta = \Gamma \cup \{\sim \mathbf{P}\}$, for a finite premise set Γ and a conclusion \mathbf{P} , then **Claim 1** tells us that

$$\text{if } \Gamma \not\vdash_{SL} \mathbf{P}, \text{ then } \Gamma \cup \{\sim \mathbf{P}\} \text{ is } SL\text{-consistent}$$

Taking the contrapositive, we get:

$$\text{if } \Gamma \cup \{\sim \mathbf{P}\} \text{ is } SL\text{-inconsistent, then } \Gamma \vdash_{SL} \mathbf{P}$$

Since $\Gamma \cup \{\sim \mathbf{P}\}$ is SL -inconsistent iff $\Gamma \models_{SL} \mathbf{P}$, this tells us that

if $\Gamma \models_{SL} \mathbf{P}$, then $\Gamma \vdash_{SL} \mathbf{P}$

That is: it tells us that the tree method for SL is complete (so long as the premise set is finite, at least).