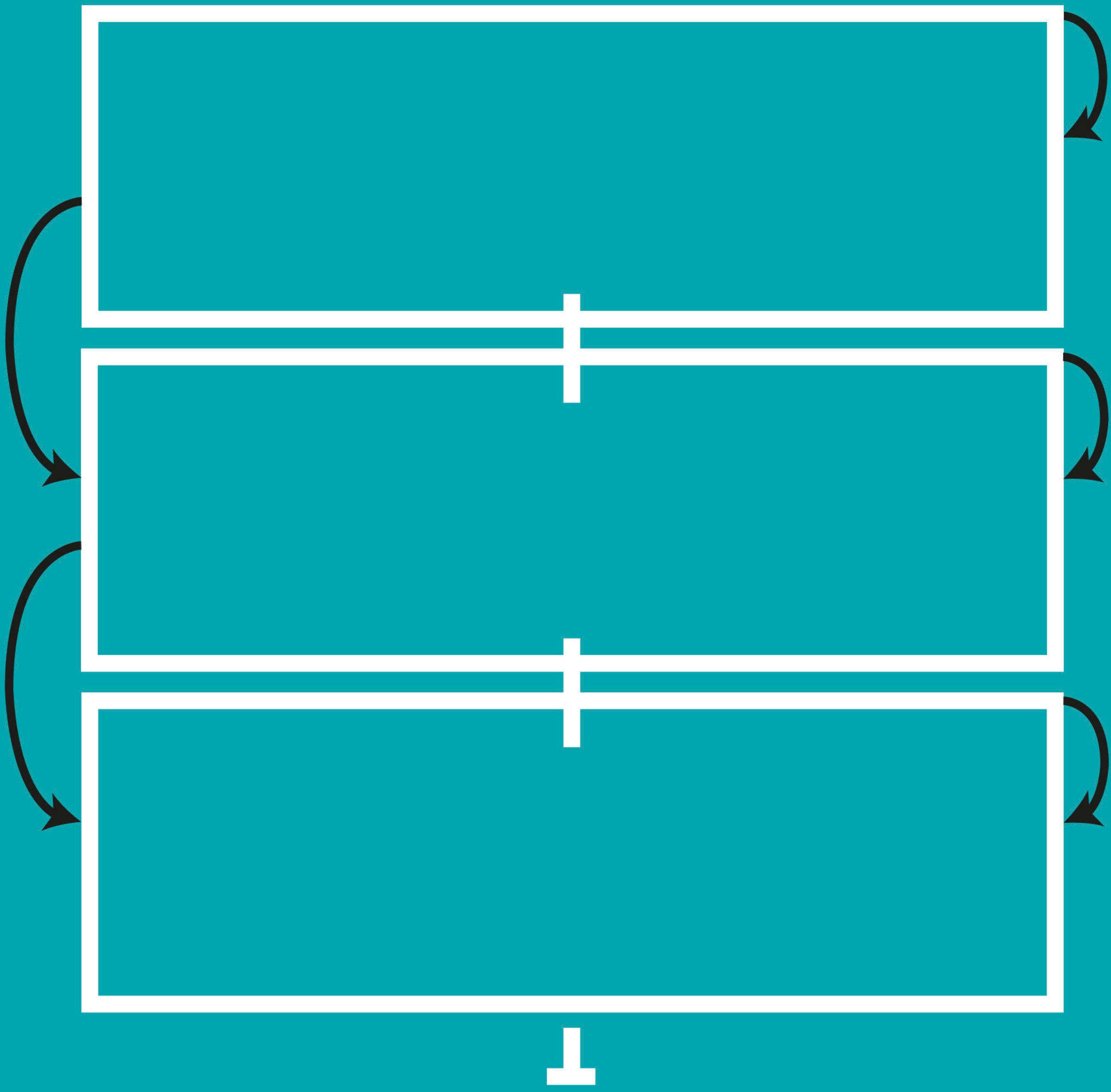
LOGIC WORKS

A Rigorous Introduction to Formal Logic

LORNE FALKENSTEIN, SCOTT STAPLEFORD,
AND MOLLY KAO



SUPPORT MATERIAL

ROUTLEDGE

Logic Works

Logic Works is a critical and extensive introduction to logic. It asks questions about why systems of logic are as they are, how they relate to ordinary language and ordinary reasoning, and what alternatives there might be to classical logical doctrines.

The book covers classical first-order logic and alternatives, including intuitionistic, free, and many-valued logic. It also considers how logical analysis can be applied to carefully represent the reasoning employed in academic and scientific work, better understand that reasoning, and identify its hidden premises. Aiming to be as much a reference work and handbook for further, independent study as a course text, it covers more material than is typically covered in an introductory course. It also covers this material at greater length and in more depth with the purpose of making it accessible to those with no prior training in logic or formal systems.

Online support material includes a detailed student solutions manual with a running commentary on all starred exercises, and a set of editable slide presentations for course lectures.

Key Features

- Introduces an unusually broad range of topics, allowing instructors to craft courses to meet a range of various objectives
- Adopts a critical attitude to certain classical doctrines, exposing students to alternative ways to answer philosophical questions about logic
- Carefully considers the ways natural language both resists and lends itself to formalization
- Makes objectual semantics for quantified logic easy, with an incremental, rule-governed approach assisted by numerous simple exercises
- Makes important metatheoretical results accessible to introductory students through a discursive presentation of those results and by using simple case studies

Lorne Falkenstein is Professor Emeritus at Western University in London, Canada.

Scott Stapleford is Professor of Philosophy at St. Thomas University in Fredericton, Canada.

Molly Kao is Assistant Professor of Philosophy at the University of Montreal, in Montreal, Canada.

"*Logic Works* is a thorough treatment of core topics in elementary logic, and of several topics in intermediate logic. Its precision and rigor is a step above typical presentations of this material. It will be an invaluable resource for teachers, as well as for students of logic who want to go beyond the basics."

Fabrizio Cariani, ***University of Maryland***

Logic Works

A Rigorous Introduction to Formal Logic

Lorne Falkenstein, Scott Stapleford, and Molly Kao

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

Cover image: Martin Shoesmith

First published 2022
by Routledge
605 Third Avenue, New York, NY 10158

and by Routledge
2 Park Square, Milton Park, Abingdon, Oxon OX14 4RN

Routledge is an imprint of the Taylor & Francis Group, an informa business

Library of Congress Cataloging-in-Publication Data
A catalog record for this title has been requested

ISBN: 978-0-367-46030-3 (hbk)
ISBN: 978-0-367-46029-7 (pbk)
ISBN: 978-1-003-02653-2 (ebk)

DOI: 10.4324/9781003026532

Typeset in Bembo
by Newgen Publishing UK

Access the Support Material: www.routledge.com/9780367460297

Contents

PART I

Sentential Logic 15

PART IV
Quantified Predicate Logic 453

Preface

Logic Works takes what might be called a philosophical approach to logic. It does not cover philosophy of logic as currently conceived, but it does ask questions about why systems of logic are as they are, how they relate to ordinary language and ordinary reasoning, and what alternatives there might be to classical logical doctrines. It covers classical first-order logic, and introduces supervaluations, many-valued logics, paraconsistent logic, intuitionistic logic, modal sentential logic, free logic, and description theories. It also covers mathematical induction, completeness and soundness proofs, and formal semantics. In addition to these field-specific topics, it considers how logical analysis might be applied to carefully represent the reasoning employed in academic and scientific work, better understand that reasoning, and identify its hidden premises. *Logic Works* deals with this material from the ground up, aiming to make it accessible to those with no prior background in logic, mathematics, or formal systems. It aims to be as much a reference work and a handbook for further, independent study as a course text. To this end, it covers more material than is typically covered in an introductory course, and it covers introductory material at greater length and in more depth. Course instructors will teach only a portion of the material presented here and will teach that portion by way of abbreviation, highlighting, and alternative explanation rather than supplementation. More advanced material is separated from foundational material by being placed in appendices, in a separate sequence of "advanced topics" chapters, and in chapter sections that build on advanced material introduced in earlier chapters. There are numerous answered exercises to facilitate self-instruction.

An outline of classical sentential logic with some nonclassical variants is completed by the close of chapter 7. Later chapters repeatedly cycle through the material originally presented over chapters 2–7, both reinforcing and embellishing that original learning. Alterations to previously presented material are highlighted, to focus attention on what is new and to make clear what is being retained with each step to a higher level.

Chapters interrupt exposition with exercises. Details are often worked out in running comments on the solutions to the selected exercises. It is necessary to work through the exercises and consult the solutions to get the full story.

Logic is learned by doing exercises. Language can be too ambiguous for the precise applications logic demands and details too small to be noticed at first can be crucial. Even the most clearly written English prose will be understood differently by different people, and the attempt to explain everything in the detail it deserves can produce an account that frustrates its own purpose by being too long and tedious. To grasp the meaning behind the language it is necessary to work on the exercises. The theory of logic is abstract. We are better at understanding concrete examples than abstract concepts. Exercises are the examples. *Logic Works* makes working through the exercises even more essential by including important portions of the expository material in the solutions to selected exercises.

Exercises that are prefaced by an asterisk (*) are answered. Answers can be downloaded from the textbook's product page at www.routledge.com/9780367460297. The answers should be used to check on the correctness of work, not as a substitute for it. Often, the solution to an exercise will make no sense to someone who has not first struggled to solve the exercise on their own. A history of failed attempts makes it clear why the answer is as it is. The exercises are progressive. Earlier ones teach things needed to handle later ones. Looking up answers to earlier exercises without working on them first makes it harder to remember the solution when needed for more advanced applications. A poor grasp of the solutions to earlier exercises makes engagement with the later exercises more difficult and time consuming. Time spent struggling with the earlier exercises is repaid by time saved dealing with the later ones.

Understanding the text is not a prerequisite for doing the exercises. Doing the exercises produces understanding. Trying to give an answer, even in the absence of understanding, is the beginning of understanding. Every second exercise is answered to serve as a prompt. An incorrect idea can be corrected by consulting a neighbouring, answered question. Neighbouring answered questions illustrate how to deal with similar cases. Sometimes only minor changes to an answered question are required to solve an unanswered one. Uncertainty over how to answer an exercise will prompt a review of important portions of the preceding text. When the text is reread with a particular difficulty in view, it will be found to say more than it did the first time through.

Instructors' Preface

Logic Works contains more material than can be taught in a single course. It does so with the aim of offering students a book that will continue to have value as a reference work and a guide for independent study. Instructors can focus on abbreviating, condensing, highlighting, sidelighting, answering questions, and taking up examples and exercises. Students can be referred to the text for more detailed explanations.

There are two sequences of chapters, a principal sequence (chapters 1–17) and an advanced sequence (chapters A-1–A-6). The advanced chapters are placed at points where they can be studied by those who have mastered the preceding principal chapters. They are not necessary for understanding the principal chapters that follow them. Later principal chapters sometimes refer to results established in earlier advanced chapters, but they do not demand understanding of how those results are established. The advanced chapters are progressive among themselves. It is necessary to have mastered the material in the earlier advanced chapters before taking up later advanced chapters.

Chapters 1, 3, 5, 7, 16, and 17 section some material off into appendices. This is done for different reasons. The appendix to chapter 1 is a summary presentation of basic concepts, better assigned for reading and reference than taken up in a class or lecture. Chapter 1 itself raises a philosophical problem with the foundations of logic and sets up an opposition between formalist and intuitionist answers to that problem that is explored throughout the subsequent chapters, and resolved (in one way and to some extent) in the advanced chapters. Otherwise, the chapter offers an introduction to methods and foundational concepts that can be quickly summarized or assigned as reading.

Instructors must decide whether to begin a course on formal logic by discussing how to formalize a natural language or by presenting a largely uninterpreted formal language. Neither approach is a happy one. The first can be confusing for many, who find that the formal language does not mesh with their intuitions about how their natural language is used. The second tries the patience of many, who do not see the point of studying a formal system that is not clearly related to their natural language. Even more unfortunately, those who are confused by the first approach are most likely to be the ones who are impatient with the second.

Logic Works begins with two brief chapters on formal syntax and semantics and only relates the formal language to English in chapter 4. The aim is to familiarize students with how the formal language works before asking them to associate it with a natural language. Relating the formal language to a natural language is very important, but it requires caution to minimize confusion and frustration. *Logic Works* is designed on the principle that introducing the formal language as a way of capturing the form of natural language sentences and demonstrations invites too many misapprehensions to be pedagogically effective. At the risk of trying students' patience, the formal language needs to first be sketched on its own terms. It can then be presented as a tool that might be used to formalize *a part* of what is said in natural languages, with due regard for the fact that the fit is not always exact.

The appendix to chapter 3 proves the expressive completeness of a formal language for sentential logic containing just five (or two or one) connectives. In passing, it introduces the concepts of disjunctive normal form and of a lean formal language. These are topics that can be skipped in an introductory course but that instructors might want to consider wedging in, depending on course objectives. The notion of a lean language is alluded to at the outset of chapter 6 to motivate introducing an abbreviated system of derivation rules. The derivation of disjunctive normal forms is discussed both in chapter 6 on derivations and chapter 7 on trees, where it serves as a hint that derivations, trees, and semantic techniques can be expected to establish the same results. The fact of expressive completeness is brought up when discussing how much of what is said in English lends itself to formalization (in chapter 4), and again in the appendix to chapter 5 when discussing what multi-valued logics aim to achieve.

The material on multi-valued logics is put in appendices to chapters 5 and 7 because it can be skipped, depending on course objectives. There are special reasons to include it in a course that aims to take a critical and philosophical approach to introductory logic. Students can find it liberating to be exposed to alternative ways in which logic might be conceived, some of which hearken back to non-Western traditions, and the alternative approaches help to strengthen their understanding of the foundations of the classical approach and of what is involved in designing an alternative system of logic.

Supervaluations are introduced in appendix 5.1. Those interested in exposing students to free logic and the philosophical problems attendant on definite descriptions and discourse concerning nonexistent objects will want to include appendix 5.1 on their course syllabi. It is foundational for sections 11.7.7 and 14.4.4.

Chapter 4, on formalization (or translation) devotes special attention to the formalization of natural language conditionals. In conformity with standard practice, it discusses the reasons why the material conditional cannot adequately formalize causal and subjunctive conditionals and presents the paradoxes of material implication. Unusually, this chapter also draws attention to special contexts where natural language conditionals are intuitively captured by material conditionals. It further offers a unique analysis of how conditionals are expressed in English and of the types of English conditionals.

Logic Works makes no attempt to "sell" the study of logic to students on the ground that it will improve their reasoning skills. In part, this is because the sales pitch is inconsistent with the appeals to intuitive foundations that are made in many chapters. In part, it is because we are sceptical of the ability of any training in logic to overcome the psychological factors that induce resistance to counterdemonstrations. But *Logic Works* does make a case that the study of logic is both intrinsically philosophically interesting and that sentential logic is instrumentally valuable for students of philosophy. Familiarity with valid and invalid sentential forms aids in the analysis of philosophical arguments. While it rarely uncovers invalid arguments, it reveals implicit premises and questionable premises, assisting in crafting rigorous, critical papers. This makes section 4.12 and the attached exercises an important component of any course for philosophy students.

Logic Works inserts chapter 4 between two chapters on formal semantics. Chapter 3 deals with the essentials of sentential semantics, as illustrated by truth tables. Chapter 5 introduces a short table method for discovering models and discusses how to establish validity and invalidity by direct appeal to the connective rules. Chapter 4 interrupts the sequence to relate the formal language to natural language sentences and demonstrations as soon as is pedagogically feasible, but also because chapter 5 can be skipped, depending on time constraints and course objectives. Courses designed to present accepted decision procedures without digressing too far to discuss what justifies those procedures will pass directly from chapter 4 to chapter 6 or 7. Chapter 5 is important for a course that aims to teach formal semantics for more advanced systems of logic (it is presupposed by chapters 9, 11, 15, and 16) and for those wanting to expose students to the techniques for demonstrating metatheoretic results (there called "informal demonstrations," and treated as such). Chapter 5 also quietly introduces natural deduction techniques in an informal context. Those planning on covering the material in chapter 6 will find that sections 5.2–3 and 5.5 prepare the way, though the material in chapter 6 does not depend on any prior knowledge of chapter 5.

Some instructors may find it preferable to pass over the material on discovering models in section 5.1 in favour of using trees. *Logic Works* opts to present trees as derivational structures, albeit structures based on semantic rules rather than intuitively consequential forms. In keeping with that approach, it treats tree models as subject to verification. As usual, the verification procedure is to climb the path, appealing to the connective rules to demonstrate how truth flows up the tree from literals to the givens. That procedure is described in section 5.1 in the context of verifying models drawn from short tables. But instructors should find little difficulty in inverting the textbook order, taking up chapter 7 in place of section 5.1, and drawing on the relevant parts of section 5.1 to explain how to verify tree models.

Chapters 6 on natural deduction derivations, and 7, on reduction trees, have been designed so that either one may be taken up in the absence of the other. Those planning to take up soundness and completeness demonstrations for more advanced systems are advised that chapters A-4, A-5, and A-6 only demonstrate the soundness of derivations and the completeness of trees. Appeal to a method for converting trees to derivations replaces the demonstration of the soundness of trees and the completeness of derivations. The reliability of that method is not rigorously demonstrated.

Chapter 6 follows up on the distinction drawn in chapter 1 between formal and intuitionistic approaches to solving the problems of logic. Derivations are there presented as a distinct approach to solving the problems of logic, grounded in intuitively obvious equivalences and entailments rather than a theory of the meaning of the sentential connectives. Chapter 5 prepared for this move by informally relying on the rules of indirect proof, conditional proof, modus ponens, and proof by cases when giving semantic demonstrations. At the outset of chapter 6, this fact is invoked to counter the prejudice that semantic theory sits in judgement of the correctness of the derivation rules. Not all instructors will agree, and whether they do or not, the observation is not offered as dogma, but as an invitation for philosophical consideration. Those who have the time to go further will find that the following advanced chapter, A-2, presents soundness and completeness demonstrations as an answer. This is an important reason for making the advanced material available. It addresses the fundamental problem exposed in chapter 1.

Though the derivation rules are justified by appeal to intuition, they are nonetheless classical. This makes recognition and discussion of alternative intuitions, prominently those labelled as "intuitionist," a topic that cannot be merely relegated to an appendix. Many courses will not have the time for this material, but it is at least there as a source for the many readers who naturally question the identification of double negatives with positives.

Those who choose to take up the study of derivations will find two differences from other treatments. *Logic Works* tends to present the more complex and restrictive derivation rules first. Learning to apply derivation rules is not like physical training. It is not necessary to start easy and build up from there. It may in fact be counterproductive. The simpler rules are in no sense components of the more complex and restrictive rules. On the contrary, giving the impression that the more complex and restrictive rules are somehow developments of the simpler and more powerful ones makes it more tempting to take liberties with the application of the more complex rules, or to imagine that they can be replaced with the simpler ones. Those who learn the more restrictive rules first are less likely to abuse them.

Secondly, *Logic Works* cautions readers that while the rules are individually intuitive, the manner in which they are to be applied to obtain an assigned result is often quite difficult to discern. The approach taken to this problem is regimented and not at all left to "intuition." *Logic Works* provides heuristics for the application of each rule. "Top-down" thinking is restricted to the application of particular rules, a "bottom-up" method is inculcated for the remainder, and a flow chart for the sequence in which rules are to be applied is followed. To mitigate indecision due to multiplicity of choice, instruction begins with a derivation system designed for a "lean language" that recognizes only atomic sentences, negations, conditionals, and the associated derivation rules. This reduces the number of rules to be juggled from 11 to five without compromising the completeness of the system. Rewriting disjunctions, conjunctions, and biconditionals as the equivalent ~/→ sentences makes for many challenging and instructive exercises, providing instruction in everything it is most important to know about doing derivations.

The lean derivation system is given a different name from the full derivation system, but chapter 6 does not teach two distinct systems. The names notwithstanding, there is one 11-rule system. It is just presented over two stages. Instructors feeling time pressure have the option of just teaching the lean system (possibly backed up with instruction on how to convert disjunctions, conjunctions, and biconditionals to ~/→ sentences, summarized in section 6.5.1). This reveals everything that is most important to learn about doing derivations. The remainder can be left to independent study. Much of the material in chapter 6.5 can also be sacrificed to time constraints. Those who do take up sections 6.5.2–4 will find that it establishes how derivations can be used to establish invalidity, by way of deriving the disjunctive normal form of the iterated conjunction of the premises and the negation of the conclusion. Anyone planning to say more about the soundness of the derivation system should consider section 6.5.6 and making the comparison with exercise 5.12. This makes it possible to assign section A-2.1 as supplementary reading for those who have studied A-1.1.

Logic Works does not aim to offer a full course in modal logic. It does aim to provide enough of an introduction to normal sentential modal logics to give readers a sense of the scope of the field, and enough background to understand Kripke semantics for intuitionistic logic and the associated tree method. Treatment of quantified modal logic is confined to an appendix to chapter A-6, which discusses theoretical and philosophical questions to the exclusion of presenting a theory. The treatment of semantics for intuitionistic logic is extended to quantified logic with identity in chapter 14, which goes out on a limb in offering an intuitionistic identity theory, echoed by the treatment of "substances" in the appendix to chapter A-6.

Somewhat idiosyncratically, *Logic Works* treats the logic of terms, predicates, and identity as a preliminary to quantificational logic, through to providing a semantics, derivations rules, and a tree method, as well as soundness and completeness results (the latter in the associated advanced topics chapter). This is done to provide a gradual introduction to the complexities of the semantics for quantified logic. It also provides an occasion to discuss definite descriptions (considered as terms) and a range of non-Russellian approaches to improper terms. Those in philosophy departments that offer courses touching on discourse concerning nonexistent objects will be interested in this material. Others can study sections 10.1–2 and 11.1–3 (and possibly 4) without taking up the following sections. The treatment of functional terms and definite descriptions in the remainder of chapters 10 and 11 takes place in two separate, supplementary modules, either of which can be studied independently of reference to the other. The module on definite descriptions gives special attention to supervaluations and to Meinongian semantics. This material is recommended for anyone going on to consider the treatment of trees for free description theory in chapter 14. Chapters 13 and 14, on derivations and trees for quantified predicate logic, place the treatment of derivations and trees for complex terms in separate subsections which can be skipped by those who chose not to study the prior material in chapters 10 and 11.

There are also two idiosyncrasies in the approach that *Logic Works* takes to quantified modal logic. One has to do with the treatment of formalization of natural language sentences in chapter 12. Many logic textbooks are oddly silent about scope ambiguities in quantified sentences of natural language. Readers of these textbooks are told that whichever unary operator happens to be mentioned first has scope over the unary operator mentioned second. Work by linguists and philosophers of language has made it clear that this is false. "There is a fork by every plate" does not have to mean that the plates are all stacked up in a pile with a single fork beside them.[1] "Someone loves everyone" is – the declarations of many textbooks notwithstanding – ambiguous. Rather than contradict those who have different intuitions about the parsing of quantified sentences, *Logic Works* identifies scope ambiguities in English and proposes policies for dealing with them. This takes some time and effort. It is necessary.

An immediate implication for those who recognize that natural languages are deeply infected with scope ambiguities is that it is hopeless to attempt to teach predicate semantics informally, by appeal to the declared meaning of natural language instantiations of formalized sentences. Many students will simply not intuit that the natural language sentence means what the instructor says it does, and they will have every right to think differently. Attempts on the part of the instructor to brow beat them into submission only leave them feeling stupid and confused and no better able to intuit the justification for the parsing the instructor insists on. This leaves no alternative but to teach the semantics for quantified logic as formal semantics. (This is one reason why *Logic Works* places chapter 3 before chapter 4. Even when doing sentential semantics it is important to establish the principle that the semantics for the formal language has to be understood independently of reference to the complex and ambiguous meanings invested in associated terms of the natural language.) The difficulty of doing this in an introductory course is not to be underestimated, and may explain why so many textbooks abandon the effort to teach modern predicate semantics in favour of recourse to syllogisms or circle diagrams or an emphasis on monadic predicate logic.

In contrast, one of the principal aims of *Logic Works*, initiated with the apparently tedious verbal semantic demonstrations of chapter 5 and steadily pursued over the ensuing semantics chapters, is to make formal semantics for quantified logic easy. This is where the second idiosyncrasy comes in. It is common in logic to distinguish between monadic and polyadic systems. This is a distinction founded on the "arity" of predicates, and it has no relation to the complexity of sentences. ∀xGxx and ∀xGax are not significantly more complex than ∀xGx. The treatment of predicate logic without quantification in chapter 11 regards all predicates as satisfied by lists of objects. In that context the added complexity that comes from dealing with lists of two or more as compared to lists of one is not worth remarking on. (This is one reason why *Logic Works* contains two preliminary chapters on unquantified predicate logic.) With that preparation, derivations, trees, and tree model semantics can all be dealt with without having to remark on a distinction between monadic and polyadic logic. Formalization is another matter. Chapter 12 deals with it by appealing to the concept of an instance to illustrate how formal language sentences must be parsed. Because the concept of an instance is syntactic, this can be done without appeal to a formal semantics. But that recourse will not serve to explain why the quantifier rules (for either derivations or trees) are sound, and it cannot explain why the quantifier rules are intuitive, which two of the four flatly are not. Failing a further appeal to instructor authority, a formal semantics is necessary.

Logic Works does this over chapters 15 and 16, the first devoted to singly quantified sentences and connective compounds of such sentences, and the second to sentences containing quantifiers with overlapping scope. This reflects a truly substantive step up in complexity, since sentences of the former sort can be dealt with by a semantics that works with variable assignments whereas those of the latter sort require appeal to variants. Both chapters provide numerous, progressive assignments training students first in the application of individual valuation rules, then in the application of sequences of those rules to establish the value of a formula on a given model, and finally in the application of the rules to discover and verify models or demonstrate that there can be no model. After this training, chapter 15 is able to conclude with two metatheorems (there simply called "principles") that lay the foundations for a soundness demonstration. Chapter 16 provides similarly careful instruction in how to step up and step down through chains of variants on variants. The appendix could have been included in the following advanced topics chapter, but is so obviously demanded by the definition of truth of a quantified sentence that it is better that it be specially selected for attachment at that point.

Logic Works does not deal with higher-order logic, undecidability, or incompleteness. However, it concludes with a brief treatment of second-order logic designed to introduce the topic in a way that conforms with the style of the earlier chapters.

Logic Works is backed up with a set of slide presentations, useful for classroom or video conference presentation, and a set of answers to ⋆ exercise questions, both available from the book's product page at www.routledge.com/9780367460297. Editable copy of the answers to the remaining exercises questions, which can be cut and pasted to reduce labour when correcting student exercises or crafting lectures, is available to instructors on request.

Sample course syllabus for a one semester, basic introduction to classical bivalent logic.

Chapter 1.1–5
Chapter 2
Chapter 3.1–4
Chapter 4
Chapter 5.1–4 (possibly replacing 5.1 with 7)
Chapter 6.1–2 and 6.6 or Chapter 7
Chapter 10.1–2
Chapter 11.1–3 (and 11.4 if Chapter 14.1–2 will be done)
Chapter 12
Chapter 13.1 and 13.2.2 or Chapter 14.1–2
Chapter 15
Other materials by choice and interest as time permits

Note

1 We owe this observation to conversations with our colleague, Robert Stainton.

Acknowledgements

Logic Works has been influenced by numerous introductory and intermediate treatments of logic, especially Beall and van Fraassen (2003); Bell, DeVidi, and Solomon (2001); Bencivenga, Lambert, and van Fraassen (1991); Bergmann, Moore, and Nelson (2014); Bostock (1997); Garson (2013); Hunter (1996); Jeffrey (1991); Kalish, Montague, and Mar (1980); Lambert and van Fraassen (1972); Nolt (1997); Prior (1962); and Sider (2010). John L Bell and Robert Stainton kindly corrected some of the more egregious errors in the treatment of intuitionistic logic and the relation between logic and ordinary language. The errors that doubtless remain are found in portions of the text they will not have seen. Jeff He read very carefully through the opening chapters and suggested a number of improvements. Special thanks to John Lehmann. We are indebted to anonymous referees for the press, whose astute observations and advice have prompted significant improvements. Very special thanks to Andrew Beck, Marc Stratton, Edward Gibbons, Lesley Hay, Kelly Winter and other members of the Taylor & Francis and Newgen production teams, who patiently accommodated a distressingly large number of corrections. Our greatest debt is to those of our students who have had the most difficulty learning logic.

References

Beall, JC and van Fraassen, Bas C, 2003, *Possibilities and Paradox,* Oxford University Press, Oxford.

Bell, John L, DeVidi, David, and Solomon, Graham, 2001, *Logical Options: An Introduction to Classical and Alternative Logics,* Broadview, Peterborough, ON.

Bencivenga, Ermanno, Lambert, Karel, and van Fraassen, Bas C, 1991, *Logic, Bivalence and Denotation*, 2nd edition, Ridgeview, Atascadero, CA.

Bergmann, Merrie, Moor, James, and Nelson, Jack, 2014, *The Logic Book*, 6th edition, McGraw-Hill, New York.

Bostock, David, 1997, *Intermediate Logic*, Clarendon Press, Oxford.

Garson, James W, 2013, *Modal Logic for Philosophers*, 2nd edition, Cambridge University Press, New York.

Hunter, Geoffrey, 1996, *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*, University of California Press, Berkeley.

Jeffrey, Richard, 1991, *Formal Logic: Its Scope and Limits*, 3rd edition, McGraw-Hill, New York.

Kalish, Donald, Montague, Richard, and Mar, Gary, 1980, *Logic: Techniques of Formal Reasoning*, 2nd edition, Harcourt Brace Jovanovich, San Diego.

Lambert, Karel and van Fraassen, Bas C, 1972, *Derivation and Counterexample,* Dickenson, Encio, CA.

Nolt, John, 1997, *Logics*, Wadsworth, Belmont, CA.

Prior, Arthur, 1962, *Formal Logic*, 2nd edition, Clarendon Press, Oxford.

Sider, Theodore, 2010, *Logic for Philosophy*, Oxford University Press, Oxford.

Symbol Summary

This summary is available to download from the book's product page at www.routledge.com/9780367460297

′	(prime)	placeholder			
^{1}	(the one)	definite description symbol			
∀	(universal)	each	α	(alpha)	assignment
A		sentence/predicate symbol	a		object name
B		sentence/predicate symbol	b		object name
C		sentence/predicate symbol	c		object name
Γ	(gamma)	set			
Δ	(delta)	secondary set	δ	(delta)	denotes
D		object domain	d		object name
𝔻	(hollow D)	outer domain			
∃	(existential)	at least one			
E		exists	e		object name
F		false	f		object name
G		sentence/predicate symbol	*f*		function variable
H		sentence/predicate symbol			
I		interpretation			
J		secondary interpretation			
K		sentence/predicate symbol	k		closed term
L		list			
M		secondary list	m		object metavariable
N		neither true nor false	n		object metavariable
Ω	(omega)	domain of worlds	o		object metavariable
P		sentence/predicate metavariable	p		name metavariable
Q		sentence/predicate metavariable	q		name metavariable
R		sentence/predicate metavariable	r		name metavariable
S		sees	s		term metavariable
T		true	t		term metavariable
			u		world
			v		world
			w		world
X		predicate variable	x		object variable
			χ	(chi)	variable variable
Y		predicate variable	y		object variable
			ψ	(psi)	variable variable
Z		predicate variable	z		object variable
			ζ	(zeta)	variable variable
⊥	(con)	contradiction symbol	∪		union
=	(identity)	is	⊢		yields
~	(tilde)	not	–|–		interderivable
&		conjunction	⊨		entails
∨		disjunction	=|=		equivalence

→	conditional	Ø		empty set
≡	biconditional	|		replace each
□	necessarily	¦		replace one or more
◇	possibly	Γ / P		demonstration
(,)	primary punctuation	↑	(nand)	not-and
[,]	secondary punctuation	↓	(nor)	not-or
{,}	set	#	(hash)	both true and false
<,>	list	?		nonevident

1 Introduction to the Study of Logic

Contents

1.1 Demonstration and Interpretation

Logic is the theory of what makes demonstration work. Metalogic is the theory of what makes logic work. This is a textbook of deductive logic, as illuminated by basic metalogic.

Demonstrations work with sentences of various sorts. Declarations, observations, commands, promises, and even questions[1] can figure in demonstration. Logic comes into play insofar as sentences have values that can be determined by the values of their parts or the values of other sentences. Paradigmatically, the sentences logic is concerned with are declarative sentences and their logically salient values are "true," "false," "neither," "both," "undetermined," and so on. These values are determined by more fundamental values ascribed to the parts of sentences. They can also be affected by larger contexts in which sentences are stated. Logic studies both how the more fundamental values determine truth and falsity (see chapters 10–11 below) and how contexts affect truth and falsity (see chapters 8–9 below). It also studies other kinds of sentences with other kinds of values – for example, "evident" and "nonevident"; "obeyed" and "disobeyed"; "upheld" and "violated"; "answerable" and "unanswerable." The question of whether the conditions of a contract (which is a type of promise) have been upheld or violated can be a matter for protracted dispute in a court of law. The same holds for the deduction of rights and duties (which are a kind of law) from other laws. Whatever else the lawyers and judges are doing, they are demonstrating what actions are required, prohibited, or allowed by contracts and legal commands. Those demonstrations are examined and systematized by logic.

A demonstration consists of a collection of sentences, called the premises of the demonstration, and a further sentence, purported to be a consequence of the premises, called the conclusion of the demonstration.

Authors and speakers will often flag what they mean to assert as a conclusion, what they mean to assert as premises, or both. Words like "therefore," "hence," "so," and "consequently," or phrases like "it follows that," and "it can be concluded that," alert the reader or the listener to expect that what is about to be stated is a conclusion. In contrast, words or phrases like "for," "since," "because," "from the fact that," "for the reason that," "given that," or "supposing that" alert the reader to expect that what is about to be stated is a premise.

In ordinary speaking and writing, premises and conclusions may be presented in any order: conclusion first, followed by premises, premises first followed by conclusion, or conclusion interspersed among premises. The demonstration may be interrupted by comments on other topics. The conclusion may be omitted if it is assumed that it will be sufficiently obvious to the audience. Important premises may be omitted, on the assumption that the audience will presuppose them. Indicator words may be omitted, on the assumption that the audience can tell

DOI: 10.4324/9781003026532-1

what the conclusion is and what its premises are. These assumptions may be mistaken. When they are, the author's intended meaning needs to be determined by questioning or, in the author's absence, reconstructed by the exercise of interpretative skills. The art of interpretation, or hermeneutics, is not strictly part of logic. Logic proceeds on the assumption that the premises and the conclusion of a demonstration have already been identified, and that all questions about what sentences mean have been resolved. However, some aspects of the art of interpretation are taken up in chapters 4, 8, 10 and 12 below.

When demonstrations are critically examined, the premises and the conclusion are isolated and numbered for ease of reference. The conclusion is marked with an indicator word or symbol and placed last, as in the following demonstration, loosely based on Sextus Empiricus, *Outlines of Pyrrhonism* 1.87. Here, the indication of the conclusion is a line placed after the last numbered premise. (In this textbook, displayed demonstrations and sentences will often be numbered for possible future reference, as with 1.1 below.)

1.1
1. Either we should have confidence in all people or only in some.
2. Having confidence in all is impossible since all are in disagreement.
3. Having confidence in some means we need to decide which ones.
4. Deciding which ones requires having a criterion for picking them out.
5. Having a criterion means putting confidence in someone's opinion of what that criterion should be.
6. But whom to have confidence in is precisely what is in question.

We are in no position to have any confidence in anyone.

Sextus did not present his reasoning in this fashion. Instead, he wrote (in the translation of Inwood and Gerson [1988: 330]),

> 1.2 Either we shall have confidence in all men or in some. But if in all we shall be undertaking impossibilities and admitting opposing statements. But if in some, let them tell us to whom we are supposed to give assent. For the Platonist will say to Plato, the Epicurean to Epicurus, and the others analogously, and so being in an undecidable conflict they will induce suspension of judgement in us.

A philosopher, seeking to analyse Sextus's demonstration, will isolate and number his premises, omit strictly irrelevant illustrative details, asides, and other interjections, supply missing premises judged from other parts of Sextus's writings to have been intended, and separate the conclusion from the premises. It is not the business of logic to do this. It is the business of hermeneutics. Once the hermeneutical task has been performed, and the premises and conclusion have been laid out, logic takes over.

This having been said, the dependence between logic and hermeneutics is mutual. One of the fundamental hermeneutical principles, the principle of charity, dictates that where there are two or more ways of understanding what someone said, the option that is most charitable, in the sense of being more likely to be true, interesting, or consequential, ought to be preferred. Logic has something to say about that.

1.2 Deductive and Inductive Demonstrations

Insofar as the conclusion of a demonstration is purported to have a value that is determined by the values of its premises, there is, as people say, "a logic" to the demonstration. That logic is concerned with whether and how the values of the premises determine the value of the conclusion.

Logic is not always concerned with determining the values of premises. Common experience and testimony, or, where that is inadequate, the investigations of the various sciences, determine such things as whether premises are true or false, or evident or nonevident, and whether promises were made, contracts signed, laws legislated, or questions asked. Logic is concerned with how contexts and the values of parts determine the values of premises, whether the value of the conclusion is indeed determined by the values of the premises, and with why it is so determined or fails to be so determined.

The relations between the premises and the conclusion of a demonstration are of two main sorts, probabilistic and deductive. Inductive demonstrations make a case for the likelihood of the conclusion. This case may be based on such things as

- experience and a supposition that things will continue to be the way they have been
- analogy with other, less controversial cases
- the measurement of frequencies or the use of sampling techniques considered to be statistically sound, and a further appeal to the laws of statistics or a theory that specifies how to calculate probability given evidence

Deductive demonstrations make a case for the certainty of the conclusion.

The study of inductive demonstrations, also called inductive logic, is not taken up in this textbook. This textbook deals with deductive demonstrations.

1.3 The Principle of Noncontradiction

Deductive demonstration is based on the principle of noncontradiction: the principle that nothing can be both affirmed and denied.[2] A deductive demonstration makes a case for the certainty of a conclusion by attempting to set up a situation in which anyone who accepts the premises of the reasoning would be forced into a contradiction were they to deny the conclusion. No reasonable, honest person wants to contradict themselves, that is, to say at one and the same time, of one and the same thing, that it both has and does not have a certain value: that it both is and is not evident; must and must not be done; will and will not be done; is and is not questionable; even that it is "both true and false" (supposing "both true and false" to be a value that some sentences can have) and not "both true and false." That is the hallmark of being illogical, duplicitous, or reckless. When presented with deductive reasoning, the honest, reasonable listener must either

- accept the conclusion
- deny one or more of the premises
- find a way to reject the logic of the reasoning, that is, to evade the charge that there is a contradiction in denying the conclusion while still accepting the premises

While the principle of noncontradiction offers a short answer to the question of what makes deductive demonstration work, there is much to be said about what is responsible for producing or preventing contradictions, particularly in less obvious cases. It helps to begin with some definitions.

A contradiction arises when the same sentence is both affirmed and denied. Classically, the sentence is a declarative sentence that is ascribed presumably incompatible values, such as "true" and "false." But incompatible values can be more generally understood to be any pair of values that should not both be attributed to the same sentence, such as "obeyed" and "ignored" for commands; "kept" and "broken" for promises; or "answerable" and "not answerable" for questions. Taken yet more generally, nonlinguistic entities can have one or other of incompatible sets of values. States of affairs can be evident or nonevident; collections and lists can contain an element or an item or not contain it; electrical switches can be on or off; objects can be the same or different, or be and not be. These incompatible states of things are the ground of the incompatible values of sentences and hence of the contradictions that arise when incompatible values are attributed to the same sentence. Logic does not stop with considering sentences to be true or false. It recognizes other values sentences can have some of which are "designated" (like "true" or "neither true nor false") as values we are particularly concerned to establish or demonstrate. More fundamentally, logic is concerned with how the values of sentences are determined by nonsentential values like "is" or "is not," and "is in" or "is not in" as said of things.

A sentence is, as the grammarians say, a series of one or more words that expresses a complete thought. Insofar as deductive logic is concerned with demonstrations that are based on the principle of noncontradiction, its focus is on those sentences that can have a value. (This concern spills over into consideration of sentences that might have more than one value, or none; and eventually spills over into consideration of how to handle sentences that have intermediate or alternative values.) A good way to identify such sentences, though not a perfectly reliable one, is by considering whether they can be denied. "It is raining," can be denied by "It is not raining"; "Eat the fruit of the tree of knowledge" by "Do not eat the fruit of the tree of knowledge"; "I will be on time" by "I will not be on time." It is not so clear that a question can be denied, but even questions can have values, like "answerable" and "not answerable" that might be attributed to them as a consequence of the values of other sentences appearing in a demonstration.

A demonstration is anything that is apparently intended by the author to convince an audience to accept a conclusion (be it by way of believing it, acting on it, obeying it, or in some other such way). A demonstration need not succeed in this enterprise. Unsuccessful attempts at giving demonstrations are still demonstrations. They are just ineffective demonstrations.

Effective demonstrations are of two sorts. Some are rhetorically effective. They have the intended effect of convincing the intended audience. Others are logically effective. They compel anyone who accepts their premises to also accept the conclusion, on pain of contradicting themselves.

It is possible for demonstrations to be both rhetorically and logically effective, or neither, or one but not the other.

Logically effective deductive demonstrations are traditionally said to be "valid." Logically ineffective deductive demonstrations are traditionally said to be "invalid."

> A demonstration is deductively *valid* if and only if anyone who affirms the premises but denies the conclusion would be caught in a contradiction.
>
> A demonstration is deductively *invalid* if and only if anyone who affirms the premises can deny the conclusion without contradicting themselves.

(The English term, "valid" comes from the Latin, "vel," which means "true," and that historical sense has never been abandoned. In ordinary language, "valid" is still used as a synonym for "true." That makes it a bad word to use to describe the relation between the premises and the conclusion of a demonstration. Truth is a value some sentences can have, not a relation between a collection of premises and a conclusion. It would be more appropriate to describe logically effective deductive demonstrations as "consequential" and logically ineffective demonstrations as "nonconsequential." This having been said, standard practice *in logic* is to use the more misleading expressions, "valid" and "invalid." That usage is followed here, with the coda that, as used by logicians, "valid" means "consequential." It does not mean "true.")

For any demonstration, there is a collection or "set" of sentences that corresponds to that demonstration: the set comprised of the premises of the demonstration and the *denial* of its conclusion. 1.1, for instance, has the following "corresponding set":

1.3 {Either we should have confidence in all people or only in some,
Having confidence in all is impossible since all are in disagreement,
Having confidence in some means we need to decide which ones,
Deciding which ones requires having a criterion for picking them out,
Having a criterion means putting confidence in someone's opinion of what that criterion should be,
Whom to have confidence in is precisely what is in question,
It is not the case that we are in no position to have any confidence in anyone}

Here, the sentence that appeared as the conclusion of the demonstration is denied, and that denial is included along with the premises to make up a set of sentences. (It is standard practice in logic to mark the start and end of a set with braces, as above.)

A demonstration is valid if and only if the corresponding set comprised of its premises and the denial of its conclusion is contradictory. This makes sense given that a demonstration is valid if and only if affirming its premises while denying its conclusion produces a contradiction.

When the sentences in a set of sentences cannot all be affirmed without contradiction the set is said to be "unsatisfiable." It cannot be "made good," so to speak.

> A set of sentences is *unsatisfiable* if and only if anyone who affirms all the sentences in the set would be caught in a contradiction.
>
> A set of sentences is *satisfiable* if and only if someone can affirm all the sentences in the set without contradicting themselves.

In applying the definitions of validity and satisfaction, it is important to distinguish between being contradictory and being false, and being noncontradictory and being true. Not everything that is false is contradictory. Many things that are false are simply contrary to fact. They could be made true, or be true in other circumstances. What is satisfiable could somehow be satisfied (that is, made to be or found to be true or evident, or to have been "obeyed" or "upheld"), even though it is not satisfied as a matter of current, local fact. What is unsatisfiable could not possibly be satisfied, where the bar for being "not possible" is placed very high: as involving a contradiction. What is contradictory is not simply contrary to fact, but contrary to itself or to other things that are accepted along with it. The sentence, "The sky is green" is false. But it is not contradictory. "The sky is green but it is not" is contradictory. It is not contradictory because it attributes both truth and falsity to "the sky is green." It is contradictory because, supposing "true" and "false" are incompatible values, it both affirms and denies that "the sky is green" has the value "true."

Similarly, the set {The sky is green, The Antarctic is in the Tropics} contains two false sentences. But there is no contradiction in affirming both of the sentences in that set.

Exercise 1.1

State whether the following sets of sentences are unsatisfiable. If they are unsatisfiable, identify one thing that anyone who believes all of them must both affirm and deny. Be careful to ensure that this thing is something that they are committed to accepting just insofar as they accept the sentences in the set and not insofar as they hold other background beliefs not described by sentences included in the set. If the sets are not unsatisfiable, describe a circumstance in which all would be true. In all exercises sets in this textbook, answers to questions marked with a star can be found online at www.routledge.com/9780367460297.

a. {Robert Walpole was British Prime Minister on 7 May 1731, Spencer Compton was British Prime Minister on 7 May 1731}
b. {Robert Walpole was British Prime Minister on 7 May 1731, Spencer Compton was British Prime Minister on 7 May 1731, Only one person was British Prime Minister on 7 May 1731}
***c.** {Robert Walpole was British Prime Minister on 7 May 1731, Spencer Compton was British Prime Minister on 7 May 1731, Only one person was British Prime Minister on 7 May 1731, Robert Walpole was not Spencer Compton}
d. {The Earth is flat, The Earth is not flat}
***e.** {The Earth is flat, The Earth is round}
f. {The Earth is flat, The Earth is round, Nothing can be both flat and round at the same time}
***g.** {The ball is either red or green, The ball is red, The ball is green}
h. {The ball is either red all over or green all over, The ball is red all over, The ball is green all over}
***i.** {The ball is either red all over or green all over, The ball is red all over, Nothing can be both red all over and green all over, The ball is green all over}
j. {You can have either ice cream or cake, You had ice cream, You can have cake}

1.4 Abstraction, Variables, and Formalization; Logical and Nonlogical Elements; Formal Contradiction

To make what is responsible for producing or preventing a contradiction stand out, logicians abstract the other aspects of a sentence or set of sentences. This is done by replacing these nonlogical elements with variables. A variable is a symbol that stands in for what can vary, that is, for any of a number of different things. In the case at hand, it stands for what can vary without affecting whether the sentence or set of sentences gives rise to a contradiction.

To take a simple case, the sentences

1.4 It is raining
1.5 It is not the case that it is raining

are contradictory. Any two sentences, one of which is the same as the other but for being prefaced by the words "it is not the case that" must be contradictory. Here, the logical element is the phrase "it is not the case that," whereas the variable element is whatever is said to both be and not be the case. The latter can be replaced with a variable. Take the capital letters of the English alphabet, A, B, C, G, H, K, to be variables for sentences ("sentence variables" for short). If more are needed, put numerical subscripts on A's to get an infinite supply, A_1, A_2, A_3, … (The other capitals are reserved for other uses.)

In the case at hand, the result is:

1.6 A
1.7 It is not the case that A

Though variables stand in the place that can be occupied by a variety of things, within any given context (within the same discussion, example, exercise, or case), no one variable can replace two or more different variants. If the variable, A, is put in the place of one occurrence of the sentence, "It is raining," it may not elsewhere be put in the place of some other sentence, such as "Pigs have wings." The two occurrences of A in the case displayed above must therefore replace the same sentence.[3] It need not be specified what sentence that is. It is allowed that it could be *any* sentence (e.g., "I promise to be on time"). But it must be the same sentence in each place the variable occurs.

Sameness is understood very strictly.

One *sentence* is the *same* as another if and only if the two sentences consist of the same words, placed in the same order.

Replacing variants with variables produces a form. In the case just given, the form is a form for a pair of contradictory sentences. This example of abstraction defines the notion of a formal contradiction.

A *formal contradiction* is a set of two sentences of the form {A, It is not the case that A}

The form contains the logical element, "it is not the case that," and the variable, A, which here stands for any sentence. Putting any deniable sentence in the place of the two occurrences of A in the form produces two contradictory sentences. (Caution needs to be exercised to ensure that when a sentence replaces a variable, the result is still a complete thought. "Is it raining?" and "Close the door!" become ungrammatical when prefaced by "It is not the case that." In some cases, an appropriate rewrite can serve instead. "Close the door" can be replaced by "The last person out is hereby commanded to close the door," which is a deniable sentence.)

There are many other contradictory forms. The sentence

1.8 It is raining but it is not raining

is by itself a contradictory sentence. Here, the logical elements are "but" and "not." The sentence that is repeated before and after these logical elements can vary. Many sentences, first stated on their own and then repeated under the word "not" and conjoined with their initial statement by the word "but," build a compound sentence that must be contradictory. Abstracting from what can vary without affecting whether a contradiction arises gives rise to

1.9 A but not A

which is one form that a self-contradictory sentence can have.

The definition of a formal contradiction can be extended to include this alternative.

A *formal contradiction* is either a set of two sentences of the form {A, It is not the case that A} or a single sentence of the form A but not-A or the form not-A but A.

This having been said, not all contradictions are formal, and hermeneutics plays a large role in identifying those that are not by showing how they can be identified with formal contradictions. As noted earlier, what people say or write can often need to be interpreted, that is, rewritten in other words to make the intention more explicit. This happens with reasoning, so it necessarily happens with the sentences that figure in reasoning. It would be absurd to insist that someone who says

1.10 The ball is red yet it isn't

has not contradicted themselves. Technically, "The ball is red" and "It isn't" are not two sentences that consist of the same words placed in the same order except for the fact that one is prefaced with the word "not" and the two are conjoined with "but." Nonetheless, this sentence is implicitly contradictory because it can be interpreted as formally contradictory. It can be rewritten as

1.11 The ball is red but [yet] it is not ['nt] the case that the ball [it] is red

Provided the rewrite accurately represents what the author meant, the original sentence counts as being implicitly contradictory.

Rewrites must be done with care. One of the fraught issues in logic concerns the scope of negations. "It is not the case that you are commanded to …" and "It is not the case that I promise to …" do not mean the same thing as "You are commanded to not …" and "I promise to not …" The account that has just been given of a formal contradiction captures what is called a "wide scope" denial. There are also "narrow scope" denials, exemplified by commands or promises to not do something. Capturing the forms of narrow scope denials is a project for later chapters. Now it only matters to be sensitive to the possibility that rewriting a sentence to replace prefatory "it is not the case that" with internal "not" needs to ensure the original meaning is preserved.

Exercise 1.2

1. *State whether the following pairs of sentences are the same or different.*
  - **★a.** Dr Jekyll is Mr Hyde. Mr Hyde is Dr Jekyll.
  - b. The coffee has sugar in it. The coffee is sugared.
  - **★c.** Either there will be a sea battle tomorrow or there will not be a sea battle tomorrow. Either there will not be a sea battle tomorrow or there will be a sea battle tomorrow.
  - d. Some doctors are lawyers. Some lawyers are doctors.
  - **★e.** Some doctors are lawyers. It is not true that no doctors are lawyers.
  - f. Some doctors are lawyers. At least one doctor is a lawyer.

2. *State whether the following sentences or pairs of sentences are formally contradictory.*
  - **★a.** Dr Jekyll is Mr Hyde. It is not the case that Mr Hyde is Dr Jekyll.
  - b. All bankrupts are despicable. No bankrupts are despicable.
  - **★c.** All bankrupts are despicable. Not all bankrupts are despicable.
  - d. All bankrupts are despicable. Some bankrupts are not despicable.
  - **★e.** It is not the case that all bats are rabid, but they are.
  - f. It is not the case that all bats are rabid, but some are.

3. *State whether the following pairs of sentences are implicitly contradictory. Justify your answers.*
  - **★a.** The light is red. The light is green.
  - b. The coffee is hot. The coffee is not hot.
  - **★c.** There are bats in the belfry. It is not the case that there are flying mammals in the belfry.
  - d. Some doctors are lawyers. Some doctors are not lawyers.
  - **★e.** Dr Jekyll is Mr Hyde. It is not the case that Mr Hyde is Dr Jekyll.
  - f. Some doctors are lawyers. No doctors are lawyers.

As noted earlier, within any one case (exercise, example, etc.) repeated occurrences of the same variable designate the same variant. However, occurrences of different variables need not designate different variants. This is part of what it means for variables to stand for what can vary. When a variable is used, one of the things it could stand for is the same thing that some other variable stands for.

Different variables certainly may stand for different things. It is always considered that they could just as well stand for different things as for the same thing. Consequently, (i) someone who wants different variables to be understood to stand for the same thing must say so; (ii) someone who wants to rule out the possibility that different variables stand for the same thing must say so; (iii) someone who says nothing either way must be understood to include both possibilities. (iii) is the default, so those in the third group are never obliged to remark on it. The reader is expected to assume that (iii) is meant when neither (i) nor (ii) is stated. Therefore,

1.6 A
1.12 It is not the case that B

and

1.13 A but not B

are not formal contradictions. This is not because A and B could not possibly stand for the same sentence, but because there is no promise that A stands for the same sentence as B.

This having been said, in important cases it is permissible to underscore the default, with phrases like, "where A and B are any two (not necessarily distinct) sentences."

1.5 A Fundamental Problem

Contradictions do not just arise when one sentence explicitly denies another, or when a single sentence contains formally contradictory components, or even when one sentence lends itself to being interpreted in a way that makes it deny another sentence. Consider the set of sentences found in exercise 1.1(c).

1.14 {Robert Walpole was British Prime Minister on 7 May 1731,
Spencer Compton was British Prime Minister on 7 May 1731,
Only one person was British Prime Minister on 7 May 1731,
Robert Walpole was not Spencer Compton}

A contradiction arises from considering all these sentences to be true. This contradiction has nothing to do with who Walpole or Compton were or how the British government worked in 1731. It arises from certain logical elements contained in these sentences. These logical elements have to do with attribution, quantification (saying how many), and identification (or its opposite, differentiation). To bring them out, use lower-case letters a, b, c, as variables for objects and use upper-case G, H, K as variables for attributes that objects might be said to have.[4] Take the variable "a" to stand for Robert Walpole, b to stand for Spencer Compton and G to stand for being British Prime Minister on 7 May 1731.[5] Then the set has the form,

1.15 {a is G,
b is G,
Only one object is G,
a is not b}

The first two sentences attribute something, G, to some objects, "a," and b. The third says how many objects have the G attribute. The fourth denies an identity, saying that object "a" is not the same object as object b. What the attribute is and what the objects are does not matter. What matters is that only one thing is said to have the attribute and yet two things that are not the same are said to have it.

When "formalized" in this way, the set is intuitively contradictory. The original English set may not have seemed so when doing exercises 1.1(a) and 1.1(b). Those exercises may have inspired the question of whether there is yet some further trick preventing 1.1(c) from being contradictory. There is not, and the formalization makes it intuitively obvious why.

But though the formalized set is intuitively contradictory, it is not formally contradictory. It does not contain two sentences, one of the form A (or "a is G," or "only one object is G," or "a is b") and one of the form not-A (or the corresponding denial of any of the other sentences mentioned). This poses a problem: What if someone does not share the intuition that this set is contradictory? What if they cannot "see" it? One recourse appeals to other things the person accepts to reduce the implicitly contradictory sentences to a formal contradiction. This amounts to demonstrating that the set really is contradictory.[6] In this case, the other things the person accepts might be ways of restating the given sentences in equivalent terms, or they might be beliefs the person has about what sentences are entailed by what other sentences.

For instance, it can be said that the sentences

1.16 a is G
1.17 b is G
1.18 a is not b

are equivalent to the sentence

1.19 At least two different objects are G

and that this entails that

1.20 It is not the case that only one object is G which does formally contradict

1.21 Only one object is G

(The difference between equivalence and entailment is that equivalence goes both ways, from what is given to what it is equivalent to, and from that back to what is given. Entailment need go only one way, from what is given to what it entails. If object "a" has attribute G and object b has attribute G and object "a" is not object b, then there must be at least two objects that have attribute G. The reverse is also the case: if at least two things have attribute G, then at least one object, "a," has attribute G, and at least one object, b, has attribute G and object "a" is not object b. In contrast, from the fact that at least two different objects are G it follows that it is not the case that at least one object is G. But it does not go the other way. If it is not the case that at least one object is G, it does not follow that at least two are. It might instead be that none are.)

Hopefully, anyone who cannot "see" why the original set is contradictory will be able to see it when the point is demonstrated by appeal to intermediate equivalences and entailments. Otherwise, a vicious regress threatens. The threat becomes apparent when considering how equivalence and entailment are defined.

> Two sentences are *equivalent* if and only if there is a contradiction in affirming either one while denying the other.
>
> Two sentences are *not equivalent* if and only if there is no contradiction in affirming one of them while denying the other.

> A sentence is *entailed* if and only if it is the conclusion of a valid demonstration, that is, if and only if there is a contradiction in denying it while affirming the premises that entail it.

There is no problem if the contradictions spoken of in these definitions are formal contradictions. Determining whether two sentences are formally contradictory is easy. It only requires looking to see if they are the same but for the fact that one of them is prefaced by "it is not the case that" or words to that effect. But when the contradictions are only implicit there is a problem. Implicit contradictions are made explicit (demonstrated to lead to formal contradictions) by appealing to equivalences and entailments. But equivalences and entailments are themselves established by appealing to the impossibility of denying them without getting caught in a contradiction. When a contradiction is only implicit, any equivalence or entailment invoked to expose it must also be only implicit, and so equally in need of demonstration. (Working only with formal contradictions never gets beyond affirming and denying the same thing. The only equivalences and entailments that are established by appeal to formal contradictions are those with forms like "A if and only if A," "A is A," and "A entails A," which are trivial. An equivalence or entailment used to establish an implicit contradiction could not be trivial and so would need its own demonstration.) A vicious regress threatens.

There are two ways to block the regress: (i) recognize that we intuitively accept certain equivalences and entailments, even though the contradiction that arises from rejecting them is only implicit, and trust that those intuitions are correct, or (ii) develop a theory of the meaning of the logical elements that explains why equivalences and entailments are contradictory. As an example of (i), it might be claimed that demonstrations like

1.22 1. a is G.
2. b is G.
3. a is not b.
―――――――――
At least two objects are G.

or

1.23 1. A.
2. If A then B.
―――――――――
B.

are just obviously valid. The contradiction in accepting the premises while denying the conclusion is so obvious that any attempt to demonstrate it from more fundamental principles would be less obvious.

Undaunted, a champion of (ii) might nonetheless attempt to explain the meaning of the logical elements involved in these sentences and attempt to show how an explicit contradiction follows from those meanings.

Both approaches to what makes logic work are explored in the chapters that follow. An application of (ii) is explored in chapters 3 and 5; one of (i) in chapter 6. Later chapters alternate between the two, and some, like the optional chapter A-2, explore the extent to which the two approaches can be trusted to deliver the same results.

It might be thought that (ii) is obviously superior to (i). But there is a problem with it. Different theories of the meaning of the logical elements can be proposed. These theories yield different results, some of which sit better with our intuitions than others. In this respect, type (ii) accounts are like geometry. There are various geometries, Euclidean and non-Euclidean, and it is a question which best describes physical space, or visual space, or tactile space. Chapters 8 and 9 show how a similar situation arises in modal logic. Even in the comparatively simple sentential logic of chapters 2–7 there are theoretical disputes. Some of them are taken up in what follows, beginning with the appendix to chapter 5. A very important issue, centred on the role of intuition in demonstration, is explored beginning in section 6.6.

In fairness, the same problem might be raised with (i). Even fundamental principles of logic such as the law of the excluded middle (either A or not-A), the reduction of double negations (if not-not-A then A), and the principle that anything follows from a contradiction might be (and have been) denied to be intuitively obvious. These concerns are also taken up, beginning with the appendix to chapter 5 and continuing with section 6.6.

1.6 Chapter Outline

In addition to using variables to stand for the nonlogical elements in a sentence or a set of sentences, logicians will often use special symbols for the logical elements. For example, the logical element, "not" or "it is not the case that," is symbolized by the tilde (~) in some logic textbooks and by the corner sign (¬) or the minus sign (–) in others. "But" and "and" are symbolized by the ampersand (&) in some textbooks and the hat (∧) or dot (•) in others. There is no standardized way of doing this. But (as this textbook is written in English), there is something to be said for using symbols that are readily accessible on the English keyboard.

A system of variables and logical symbols constitutes a formal language, that is, a language used to represent the forms of sentences and sets of sentences of a natural language. An increasingly enhanced formal language for demonstrative logic is developed over the chapters that follow.

A language for formalizing some basic deductive demonstrations is presented in chapter 2. In chapters 3 and 5, an account is given of why forms described using this system are equivalent or not equivalent, and contradictory or noncontradictory. In chapters 5, 6, and 7, procedures for testing for validity, unsatisfiability, entailment, and equivalence are presented. In the optional chapters A-2 and A-3, these test procedures are shown to agree.

Initially, the formal language is presented on its own terms. Its symbols are only incidentally related to ordinary language. The relation between the formal language and English is discussed in chapter 4, after the workings of the formal language have been discussed in chapters 2 and 3. This may seem disorienting, but it is important to bear with it. The formal language is a power tool, designed for application to the materials provided by natural languages. Like the power tools of the construction trade, its use and limitations need to be understood before it is applied, to avoid damage and injury.

Procedural Note

It is natural to attempt to understand what is unfamiliar (the formal language) by analogy with what is familiar (the natural language), just as it is natural to think that the way to learn to use a power saw is to pick it up and start hacking away at something. An early attempt to relate formal languages to sentences and demonstrations in ordinary language can cause more confusion than illumination. The correspondence between logical systems and natural languages is not exact, in part because natural languages carry elements of meaning that basic logical systems are not designed to formalize, and in part because expressions in natural languages need to be understood in context. Those beginning the study of logic know their natural language better than they know the formal language. When they are presented with natural language interpretations of the symbols in a basic formal system, they invest the symbols with too much meaning. This causes them to misapply the formal tool, producing material damage. When the material damage is discovered, it produces upset and confusion, which are cognitive injuries. Proper advance training in the workings of the formal language can avoid both.

In chapter 4, it is shown that there are limitations to the formal language that has been developed over chapters 2–3. Some of the limitations serve as the occasion for the presentation of refinements to the system. This is done repeatedly in chapters 8–17, with the introduction of formal languages of increasing complexity, each adding to what was done by the previous one to extend the system to cases it previously could not handle.

At various points, the textbook presentation of this material is interrupted with chapters on advanced topics, labelled A-1 through A-6. These chapters are inserted at the point where enough has been said to understand the material taken up in the chapter, but they can be skipped or reserved for a more advanced course or for independent study. Results proven in the advanced chapters are often mentioned in subsequent chapters, but it is in no case necessary to understand how those results were proven. The material contained in appendices is also optional.

Technical Appendix: Elements of a Theory of Demonstrative Logic

The following concepts are fundamental and common to all the languages that will be studied. Later chapters will make them more familiar and provide opportunity to become experienced in their application. They need not be memorized or fully understood at this point. This appendix is intended as a useful summary and reference for later work.

A *sentence* is any sequence of one or more symbols that the language in question recognizes as a sentence. As such a sentence must:

(i) consist of elements included in the vocabulary of the language
(ii) list those elements in an order approved by the grammar of the language.

Two sentences are the same if and only if they consist of the same symbols, placed in the same order.

Two sentences are *different* if and only if they are not the same.

Two sentences are *opposite* if and only if they are the same but for the fact that one of them is preceded by "it is not the case that," or other words or phrases to that effect, or symbols recognized by the language in question as symbolizing words or phrases to that effect.

In addition to sentences, all systems of logic are concerned with three main groups of sentences: sets, lists, and demonstrations. They are also concerned with a special relation that can be defined in terms of sets and lists, that of a function.

(1) A *set* consists of 0 or more *different* members (for now, the set members are sentences), collected in no particular order.

A set with no members in it, called the empty set, counts as a set, as does a set with only one member in it, called a unit set.

By convention, sets are identified by being enclosed in braces with the members of the set separated from one another by commas. In the case where the set contains only one member, and context makes it clear that the member is being considered as the member of a unit set, the braces may be omitted. The empty set may be designated either by { } or by the symbol, Ø, or simply by a blank space in a place where a set would otherwise be identified. When the intention is to speak about some set or other, without specifying what members it contains, or without having any one set in mind, the Greek capital, Γ (gamma), is used. (Γ is L flipped, and L is later used to stand for a related concept, that of a list. Some flipped symbols [$\perp$, Γ, $\forall$, $\exists$] are so pleasingly evocative that they are used in this text even though they are not readily accessible on the keyboard.) On occasions where more than one set must be referred to, numerical subscripts or primes (′) may be employed, as below. On some (rare) occasions, the Greek capital, Δ (delta), may also be used.

A further symbol, $\cup$ (union), is used to represent the set that results from combining two sets or from adding a sentence to a set. $\Gamma \cup \Gamma'$ is the set of everything in Γ supplemented with everything in Γ'. $\Gamma \cup A$ is the set of everything in Γ with the addition of A. Some writers will insist that only sets may be united and hence that $\Gamma \cup A$ should be written $\Gamma \cup \{A\}$, where {A} is the unit set containing A. In keeping with a general policy to reduce clutter whenever no mistake can arise from doing so, *Logic Works* is not so scrupulous. Since $\cup$ is understood to be preceded by a set, braces may even be omitted on the left when the set is a unit set. However, they are retained to delimit the contents of sets that contain more than one member.

Two sets are *different* if and only if there is at least one member that is in one set but not in the other; otherwise they are *the same.*

One set, Γ, is a *subset* of another, Γ′, if and only if every member of Γ is in Γ′. There may, but need not be members of Γ′ that are not in Γ (so every set is a subset of itself). As a special case, Ø is a subset of every set.

One set, Γ, is a *superset* of another, Γ′, if and only if every member of Γ′ is in Γ. There may, but need not be members of Γ that are not in Γ′ (so every set is also a superset of itself).

Sets can themselves be members of sets. There can be sets of sets, sets of sets of sets, and so on.

(2) A *list* consists of 0 or more *not necessarily different* items (for now, the list items are sentences) listed one after another. By convention, lists are identified by being enclosed in angle brackets with the items on the list separated from one another by commas. The ordinary sense of the term notwithstanding, a list of no items counts as a list, as does a list of just one item. In the case where the list consists of a single item, the angle brackets may be omitted. The empty list is designated by < > or by Ø or simply by a blank space. On lists, it is often only the last item on the list that matters. In that case, the English capital, L, is used to designate whatever earlier items there might be on the list. Since the appearance of L by itself indicates the presence of a list, the angle brackets may be omitted when L is used. For example, L,A designates a list that begins with the items on the list, L, and ends with A.

In addition to standing for the earlier items on a list, L may be used when the intention is to speak of some list or other, without specifying what items it contains, or what item it ends with. On the rare occasions where more than one such list must be referred to, numerical subscripts or primes (′) may be employed. On some occasions, M may also be used to designate a second list.

Two lists are *different* if and only if the same items do not occur in the same order on each list; two lists are *the same* if and only if they consist of the same items listed in the same order.

Both conditions must be satisfied; a list, L, that consists of the same items found on another list, M, but that does not list those items in the same order as they are on M is not the same as M.

Lists can themselves be items on lists, as can sets. There can be lists of lists, lists of sets, lists of lists and sets, and so on. There can also be sets of lists.

Both sets and lists might metaphorically be described as bags of members or items. The difference is that a set is like a bean bag whereas a list is like an egg carton. The members of a set are just there, in no order, whereas each item has its own place on a list. On a list, the same item can occur two or more times in different places, whereas in a set this makes no sense. The set {1,1,2} is no different from the set {1,2,1} or from the set {2,1} since what defines a set is just what members it contains. But on a list, the list items have place relative to one another, which makes it possible for the same item to occur more than once, in different places. The lists, <1,1,2>, <1,2,1>, <1,2>, and <2,1> are all different from one another. Some of these lists contain three items, others two, and some contain the same number of the same items but differ from one another in how the items are ordered on the list.

In the examples just given, numbers make up the set members and list items. But anything (and for present purposes notably sentences) can appear in a set or on a list, as suggested by the example of a task list or a bag of groceries. Numerals are often used as names for list items or set members that it would be irksome to have to describe or name in other ways. When numerals appear in examples in *Logic Works*, they only rarely stand for mathematical objects. They most often designate list items or set members that are not worth identifying beyond being called "member/item 1," "member/item 2," and so on.

(3) A *demonstration* is a list, <Γ,A>, consisting of a set of 0 or more sentences, Γ, followed by a further sentence, A. The sentences in Γ are called the premises of the demonstration and A is called its conclusion.

Notwithstanding the oddity of doing so, a demonstration with no premises is recognized as a demonstration.

Γ is a (possibly empty) set, not a list. The list is the list of the set, Γ, followed by the sentence, A.

As noted in section 1.1 above, one form for presenting a demonstration assigns numbers to a finite subset of the sentences in Γ and lists these numbered sentences above a horizontal line, below which the sentence A is placed. *Logic Works* also uses the briefer notation, Γ / A, to represent demonstrations. When Γ is empty, / A by itself indicates that A is the conclusion of a demonstration that has no premises. (Presumably, A is such a good conclusion that it is valid all by itself.) Where appropriate, Γ may have the appearance of a set of sentences separated from one another by commas. The forward slash (/) is used in place of a comma between the last sentence in Γ and A. It serves as a conclusion indicator. Because the forward slash by itself indicates that what lies to the left is a set of premises of a demonstration and what lies to the right is a conclusion, the angle brackets and the braces are omitted.

When Γ / A is valid, A is said to be *entailed* by Γ. When Γ / A is invalid, Γ does not entail A. A special symbol, the double turnstile (⊨) is used to represent entailment. A slash through the double turnstile (which for purposes of work at the keyboard can be approximated by the bar symbol followed by the not equals symbol, |≠) represents nonentailment. Γ ⊨ A is read as "gamma entails A," Γ ⊭ A as "gamma does not entail A." Like /, ⊨ is understood to be preceded by a set and followed by a sentence. When Γ is a unit set, the braces may be omitted.

When Γ / A is valid, the corresponding set, Γ ∪ not-A, is unsatisfiable. When Γ / A is invalid, Γ ∪ not-A is satisfiable.

(4) A *function* is a special relation between a set of lists, called the *arguments* of the function, and another set, called the *range* of the function. The lists in the first set must all have the same length, and they cannot be empty. If the lists are lists of one thing, the function is called a one-place function, if they are lists of two things, the function is called a two-place function, and so on. The second set, the range, is a set of one or more *values*.

Functions are commonly notated in the form *f*(L) is *x*, where *f* is a symbol used to designate the function, L is an argument (a list), and *x* is a value drawn from the range. Though the argument is always a list, even if only a list of one, it is customary to put parentheses around it, rather than angle brackets. Functions are ubiquitous in mathematics and mathematicians like to use = in place of "is." An example drawn from mathematics is +(1,2)=3, read as "the sum of 1 and 2 is 3," where + is the "sum of" function. (In the case of two-place functions, it is common to put the function between the list items. This produces the more common 1 + 2 = 3.) An example drawn from everyday life is *f*(Alma) is Boda, read as "the mother of Alma is Boda," where *f* is the "mother of" function, "Alma" is one of the arguments for that function, and Boda (who is, incidentally, Alma's mother) is the value that argument takes under that function.

The set of lists comprising the arguments of a function is based on a more remote set, called the *domain* of the function. (When the domain and the range are discussed together, the range is often called the *co-domain* instead.) The domain specifies what objects are to be used to make lists. For each function of *n* places (where *n* may be 1, 2, 3, or whatever number), the set of lists must include every list of *n* objects that it is possible to form from objects in the domain. When the function is a one-place function, the set of argument lists is just the set of each object in the domain. When it is a two-place function the set of argument lists contains each ordered pair that it is possible to form using (not necessarily distinct) members of the domain.

For example, consider a domain that has two members. It does not matter what the members are, so just consider the domain to be {A, B} where A and B are labels for these two members. Consider also a co-domain or range of values, say the set {T, F}. (In principle, the co-domain might include some or all of the members of the domain or it might include none. The aim at this point is to cover all bases, so nothing hangs on what is considered to be in the domain or what is considered to be in the co-domain.) Now consider a three-place function that relates (or "maps") this domain to (or onto) these values. The function must assign exactly one value from the set of values in the range to each list of three that it is possible to form from members of the domain. In the case at hand, this means that it must make an assignment of exactly one of T or F to each of the following lists:

1.24 <A,A,A>
<A,A,B>
<A,B,A>
<A,B,B>
<B,A,A>
<B,A,B>
<B,B,A>
<B,B,B>

Each of these lists is an argument that the function uses to determine a value. Given the domain and the values specified, there will be as many different three-place functions as there are different ways of assigning exactly one value from the range to each list. Here is a list of the values assigned to each of these lists by a function that can be symbolized as, 1/a, and that is defined as the function that assigns T to each argument that contains exactly one A and otherwise assigns F.

1.25 1/a(A,A,A) is F
1/a(A,A,B) is F
1/a(A,B,A) is F
1/a(A,B,B) is T
1/a(B,A,A) is F
1/a(B,A,B) is T
1/a(B,B,A) is T
1/a(B,B,B) is F

There are of course many other ways of assigning exactly one of T or F to each list in 1.24. Each of them is a different function, beginning with the function that assigns T to each list and F to none and continuing through a total of 256 different ways of assigning exactly one of T and F to each of these eight lists. Were A, B, and C considered to be sentence variables and T and F to designate the values, true and false, these functions would represent various ways of assigning a single truth value to each different list of the three sentences in the domain.

Exercise 1.3

- ***a.** Specify a possible domain and a range for the function, "square of."
- ***b.** Specify a possible domain and a range for the function, "mother of."
- ***c.** Why is "square root of" not functional over the domain of rational numbers and the range of real numbers?
- ***d.** Why is "brother of" not functional over the domain and range of human beings?
- ***e.** Specify a domain and a range over which the relation "spouse of" is functional.

Notes

1 Questions can set the task for reasoning or challenge it. The answerability of questions can be a topic for reasoning. Hamami and Roelofsen (2015) offer introductory comments on the logic of questions.

2 Immanuel Kant (1724–1804) identified the principle of noncontradiction as "the highest principle of all analytic judgments," which for Kant are sentences asserting the containment of predicate concepts under subject concepts. For Kant, contradictions accordingly affirm and deny that one thing falls under a concept. On this account, contradiction has more to do with relations of containment under a concept than truth and falsity. Truth and falsity are derivative from the more fundamental relation of being in or not in. Arthur (2011: 26) traces the thesis that deductive reasoning is based on "incompatibility" back to the Stoic logician Chrysippus (279–206 BCE).

3 Those who are concerned to flag cases where a term is mentioned might want to see A put in quotes. *Logic Works* is not so scrupulous. It does not employ any special means for distinguishing between *use* and *mention*, except in those cases where a reader might initially confuse an expression that is only being mentioned with one that is being used. In that case the mentioned term is put in double quotes.

4 There is a shift here between using variables to stand for linguistic entities (sentences) and using them to stand for non-linguistic entities (objects and predicates of objects). This is in order. The shift is discussed in chapter 10.

5 The use of "a" as a variable has a drawback. It is both a lower-case letter of the English alphabet and a word of English (an indefinite article). Consider, for instance the three stand-alone occurrences of the first lower-case letter of the English alphabet in the sentence, "The use of a as a variable has a drawback." In many contexts, inserting the variable in a sentence without further ado could lead a reader to confuse it with the definite article at first glance, forcing a double-take when the rest of the sentence does not fit with that assumption. Throughout this work occurrences of "a" will often be quoted to ease the reading when the letter is used as a variable. This caution is not necessary with other variables.

6 The implication is noteworthy: demonstrations are only called for when intuition fails us.

References

Arthur, Richard TW, 2011, *Natural Deduction*, Broadview, Peterborough, ON.
Hamami, Yacin and Roelofsen, Floris, 2015, "Logics of Questions," *Synthese* 192: 1581–4.
Inwood, Brad and Gerson, LP, 1988, *Hellenistic Philosophy: Introductory Readings*, Hackett, Indianapolis.

Part I

Sentential Logic

2 Vocabulary and Syntax

Contents

2.1 Introduction

Deductive logic seeks to identify the factors responsible for producing or preventing a contradiction. This is done by constructing a formal language, that is, a language that contains variables to stand for the nonlogical parts of sentences or sets of sentences, and special symbols to stand for the logical parts. The nonlogical parts are the parts that can vary without creating or eliminating a contradiction. The logical parts are the parts that play a role in creating or preventing a contradiction.

Formal languages can be constructed at various levels of specificity, incorporating more or fewer of the elements responsible for giving rise to contradictions. It is best to start with fewer. This and the following chapters start with those elements that give rise to contradictions when sentences are compounded with one another, be it in demonstrations, sets, or compound sentences. The study of these elements comprises the logic of sentences, also called sentential logic.

Some sentences are atomic, which is to say that they do not consist of parts that are themselves sentences. "Pigs have wings" is an atomic sentence. Other sentences are built up from atomic sentences using connective expressions.

A connective may be unary, in which case it is attached to one sentence. "It is not the case that pigs have wings" is a compound sentence, compounded from the atomic sentence "Pigs have wings" and the unary connective "It is not the case that …"

A binary connective connects two sentences. "Pigs have wings but they cannot fly" is a compound sentence, compounded from the atomic sentence "Pigs have wings" and a further sentence "Pigs cannot fly," using the connective expression "but." In this example, "but" connects an atomic sentence with a sentence that is already a compound of an atomic sentence, "Pigs can fly," and a unary connective, "not." "But" is nonetheless still a binary connective because it connects only two sentences, even though one of them is itself compound.

There are also connective expressions that are used to connect three or more sentences. "At least one but no more than two of Alma, Boda, and Crumb were advised by Gear" uses the higher-place connective "at least one but

DOI: 10.4324/9781003026532-3

no more than two of …" to connect the sentences "Alma was advised by Gear," "Boda was advised by Gear," and "Crumb was advised by Gear."

Exercise 2.1

State whether the following sentences are atomic or compound. If they are compound, identify their atomic components. An atomic component of a compound sentence must be a part that is a sentence in its own right. As a special case, some compound sentences may consist of one atomic component and a connective expression.

***a.** Active power is what enables someone to bring about an effect.
b. It is the soul that sees, and not the eye; and it does not see directly, but only by means of the brain.
***c.** Bodies produce ideas in us by impulse, as that is the only way we can conceive bodies to operate in.
d. Our ideas are adequate representations of the most minute parts of extension.
***e.** If morality has some influence on human actions, it is right to try to inculcate it; and that multitude of rules, with which all moral writings abound, is not pointless.
f. A succession of ideas constitutes time, and is not only the sensible measure of time.

Some connective expressions are of interest to logicians and others are not. Most systems of sentential logic include symbols for five connectives, though systems vary in which symbols are used for these five connectives.

There are more than five connectives that are of interest to logicians and many more that are not of interest to logicians. The question of how to identify and deal with all the logically interesting ones is taken up in the appendices to chapters 3 and 5.

Some systems of sentential logic, including the one presented here, also use a special symbol, ⊥, to stand for contradiction. This is appropriate given the central role of contradiction in deductive logic (see chapters 1.4–1.6). ⊥ has various names: "up tack," "falsum," and "bottom" or "bot" are some. "Con" is a memorable name that alludes to the precise meaning the symbol has in this textbook while avoiding troublesome associations with a sentential value and irrelevant associations with its shape. The symbol is best entered at the English keyboard with "space," "capital I," "space," all underlined: $\underline{\ I\ }$.

⊥ and symbols for the five chosen connectives make up the logical vocabulary of a formal language for sentential logic, called SL. The language is further outfitted with an infinite stock of variables, to stand for sentences, and a pair of punctuation marks.

Vocabulary of SL

(The commas that appear below are not included in the vocabulary; they serve to separate those items that are.)

Sentence letters: A, B, C, G, H, K, A_1, A_2, A_3, …
Contradiction symbol: ⊥
Connectives: ~, &, ∨, →, ≡
Punctuation marks:), (

The infinity of the stock of sentence letters is provided for by placing numerical subscripts on A's. The subscripts are considered part of the A's they are attached to, so numerals do not need to be added as separate vocabulary items. The six unscripted capitals are unnecessary. They are included to make the language easier to read and use. Six sentence letters are more than are needed for most purposes. There is something to be said for working with a logical vocabulary that uses characters that are readily accessible on the keyboard, thereby facilitating electronic submission of work. Ideally, those characters would also be mnemonic. Using just six English capitals as sentence letters frees the remainder for other purposes. (D, E, F, and I, for instance, are later used for "domain," "exists," "false," and "interpretation." J is reserved for a second interpretation in those cases where more than one interpretation is under consideration.) When this approach is not always followed, it is out of respect for conventions, because the symbol is only infrequently used in student exercises, because the best mnemonic English character has already been put to other uses, or because the symbol is especially evocative of what it stands for.

The names of the connective symbols are tilde (~), ampersand (&), wedge (∨), arrow (→), and triple bar (≡). At the keyboard, arrow can be produced with "hyphen," "right angle bracket": ->. Triple bar should be rendered with "left angle bracket," "hyphen," "right angle bracket": <->. The names for the connectives are intentionally descriptive only of the shape, not of the meaning. At this point, only the shape matters. Meaning is taken up in chapter 3.

When the sentence letters of SL are interpreted by having sentences of English or other natural languages assigned to them, and natural language connective expressions are assigned to the connectives of SL, SL can be used to represent the forms of sentences, sets of sentences, and demonstrations of natural languages. The use of SL to formalize English sentences, and the techniques for instantiating sentences of SL in English (coming up with an English sentence that is an instance of the form described by a sentence of SL) are taken up in chapter 4.

The remainder of this chapter deals just with how the vocabulary elements of SL are arranged.

Any vocabulary element or sequence of vocabulary elements constitutes an expression of SL.

In this and the preceding chapter, some symbols are introduced that do not appear in the vocabulary of SL (P, Q, R, Ø, Γ, Δ, L, M, ∪, /, ⊨, ⊢, {, }, <, >). On the following pages, these symbols are often mixed with vocabulary elements of SL. The resulting formulas are not expressions of SL. Expressions of SL consist only of the vocabulary elements listed earlier.

Even lower-case letters of the English alphabet, or upper-case letters in boldface, italics, or other fonts, or upper-case letters other than A, B, C, G, H, and K in the font used here, or upper-case B, C, G, H, and K when followed by numerical subscripts are not vocabulary elements of SL.

Ellipses and commas are also not vocabulary elements, even though they may be mixed in with vocabulary elements, as in the description of the vocabulary of SL given above. There, the ellipses indicate that "A" may have numerical subscripts that go to infinity. Commas are used to separate listed items from one another.

Exercise 2.2

State whether the following symbol sequences are expressions of SL. *Justify your answers.*

*a. ~⊥
b. [A ∨ B]
c. ⊥ A
*d. ((A ≡ B_7()
e. ~)A & B(
f. ¬ A ∧ B
g. (***A*** ∨ C)
h. (A ≡ (D & C))

While any vocabulary element or sequence of vocabulary elements is an expression of SL, not just any vocabulary element or sequence of vocabulary elements is a sentence. To be a sentence, the vocabulary elements need to be arranged in the right way. This way is defined by a brief system of formation rules. The rule system is recursive, which means that longer sentences can be built by reapplying the rules to sentences constructed by previous applications of the rules.

According to the rules, ⊥ is a sentence and each of the sentence letters is a sentence. The rules further specify that the result of putting ~ in front of a sentence is a sentence. For example, because ⊥ is a sentence, ~⊥ is a sentence, and thus so are ~~⊥ and ~~~⊥. The rules finally specify that taking any two sentences, putting a binary connective between them, and enclosing the result in parentheses produces a sentence. Because A and ~A are sentences, (A & ~A) is a sentence. And because rules may be applied on top of rules, the fact that ~⊥ and (A & ~A) are sentences means that ~(A & ~A) and (~(A & ~A) ∨ ~⊥) are sentences.

Meaning is not at issue at this point. Only form and formation are at issue. It does not matter whether arranging vocabulary elements in accord with the formation rules produces a form that might be taken to mean something absurd or contradictory. On the contrary, it would be a bad thing if the formation rules of SL prohibited constructing a formally contradictory sentence, since SL is supposed to identify some of the factors responsible for generating contradictions. What is invalid or unsatisfiable needs as much to be symbolized as what is valid or satisfiable.

The formation rules constitute the grammar or, in more technical language, the syntax of the language.

The syntax for SL can be more precisely formulated by using the symbols P and Q (and later, R and numerically subscripted P's) as variables for expressions of SL. Because SL is itself a language built around variables that stand for sentences of a natural language, this makes P and Q variables for variables, or what are called metavariables.

Syntax of SL

An *expression* is any vocabulary element or sequence of vocabulary elements.
A *sentence* is any expression formed in accord with the following rules:

(SL): If P is ⊥ or a sentence letter, then P is a sentence.
(~): If P is a sentence, then ~P is a sentence.
(bc): If P and Q are two (not necessarily distinct) sentences, then (P & Q), (P ∨ Q), (P → Q), and (P ≡ Q) are sentences.
(exclusion): Nothing is a sentence unless it has been formed by one or more applications of the preceding rules.

In what follows, the expression (~) is used to name the second of the formation rules given above. It is not used as an expression of SL. Similarly (SL), (bc), and (exclusion) are names of the first, third, and fourth of the formation rules.

According to the rules, a tilde may only be added to the front of a sentence, a binary connective may only be placed between two sentences, and punctuation marks may only be used in left and right pairs, placed on either side of the sentences conjoined by a binary connective. Punctuation is never placed around ⊥ or the sentence letters, and never used when adding a tilde to a sentence. ~(A) is not a sentence. ~(~A) is not a sentence. (A & (~B)) is not a sentence. And so on. The only rule that adds punctuation is (bc). For each pair of corresponding parentheses, there must be exactly one binary connective that is dedicated to that pair of parentheses. Where there is no corresponding binary connective there should be no parentheses. Where there is a binary connective there must be exactly one pair of parentheses. No pair of parentheses may be shared by more than one binary connective, no binary connective may use more than one pair of parentheses, and no pair of parentheses may appear that is not dedicated to exactly one binary connective.

Exercise 2.3

State whether the following expressions are sentences of SL. *Justify your answers, following the example of the answers linked to the ★ questions. Be careful to appeal to (exclusion) when justifying the claim that an expression is not a sentence.*

★a. AB
b. A~B
★c. (A & (~B))
d. (A & ~B)
★e. (A & ~(~B))
f. (A & ~(~B ∨ C))
★g. (A & A)
h. (A & ~A)
★i. ((~⊥) & B)
j. ~(⊥ & B)
★k. ~(~A & B)
l. ((A & ~(B ∨ C))
★m. ((A & B & C) ∨ ⊥)
n. ((A & B) & C ∨ ⊥)
o. (((A & B) & C) ∨ ⊥)
p. (⊥ → ~⊥)
★q. (~((~A → A) ≡ ~(A & ~A)) ∨ (~A → A))
r. (A ∨ (~(B ∨ C)))

Understanding the syntactic rules requires an ability to relate expressions containing the metavariables P, Q, and R to sentences of SL that instantiate those expressions. Expressions containing metavariables, like P, ~P, and P & Q, have the status of forms that various sentences of SL can share. Relating metalinguistic expressions and sentences of SL means recognizing which sentences of SL are instances of which forms.

> An *instance* is any result of replacing the variables in a form with objects that the variables could stand for, doing so in such a way that the same object is put in the place of each occurrence of the same variable.

Instances are not always direct. ~(A & B) is an instance of ~(P & Q), but so are:

2.1 ~(A & A)
2.2 ~(⊥ & A)
2.3 ~(~A & B)
2.4 ~(A & ~B)
2.5 ~(~A & ~B)
2.6 ~(((~B → ~G) ∨ K) & ~(B ≡ ~K))

Like all variables, P and Q could stand for arbitrarily complex sentences of SL. They could also stand for the same sentence of SL.

Exercise 2.4

1. *State whether the following sentences are instances of the form,* ~P. *If they are, identify the sentence that instantiates* P. *If they are not, say why not.*

★a. ~⊥
b. ~~B
★c. (~C & ~G)
d. ~(C & ~G)
★e. ~((~A & B) ∨ C)
f. (~(~A & B) ∨ C)

2. *State whether the following sentences are instances of the form,* (P & Q). *If they are, identify the sentences that instantiate* P *and* Q. *If they are not, say why not.*

★a. (A & A)
b. (⊥ & B)
★c. ((A & B) & ⊥)
d. (A & ~A)
★e. ~(A & B)
f. (~A & B)
★g. ((~A & B) ≡ C)
h. (~A & (B ≡ C))

3. *State whether the following sentences are instances of the form,* ~(P → ~Q). *If they are, identify the sentences that instantiate* P *and* Q. *If they are not, say why not.*

★a. ~(~A → ~B)
b. ~(A → ~A)
★c. (~A → ~⊥)
d. ~(~A → ~~⊥)
★e. ~(A → ~(B → C))
f. ~(A → (~B → C))

4. *State whether the following sentences are instances of the form,* (~P → Q). *If they are, identify the sentences that instantiate* P *and* Q. *If they are not, say why not.*

★a. ~(A → B)
b. (~A → ~B)
★c. (~~A → B)
d. (~(~A → B) → ~(C → A))

★e. $(\sim\bot \rightarrow \bot)$
f. $((\sim A \rightarrow B) \equiv C)$

5. *State whether the following sentences are instances of the form,* $(P \rightarrow (Q \vee \sim P))$. *If they are, identify the sentences that instantiate* P *and* Q. *If they are not, say why not.*
★a. $(A \rightarrow (A \vee \sim A))$
b. $(A \rightarrow (\sim A \vee \sim A))$
★c. $(\sim A \rightarrow (B \vee \sim A))$
d. $(\sim A \rightarrow (B \vee \sim\sim A))$
★e. $(\sim(A \vee B) \rightarrow (B \vee \sim\sim(A \vee B)))$
f. $(\sim B \rightarrow (A \vee B))$

6. *(This question draws on material presented in chapter 1.3 and point 1 of the technical appendix to chapter 1. Please review those sections before proceeding and consult the solutions to the ★ questions before attempting those that are unanswered. This having been said, the solutions demonstrate that it is possible to answer these questions without knowing anything about what* Γ, *"set,"* $\cup$, *or* { } *stand for. Forms can be instantiated without knowing anything about the meanings of the elements used to construct either the form or the items plugged into it. This is the most valuable lesson to take away from this exercise.)*

State whether the following are instances of the form, $\Gamma \cup \sim P$, *where* Γ *is a set of sentences and* P *is a sentence. If they are, identify the set of sentences that instantiates* Γ *and the sentence that instantiates* P. *If they are not, say why not. Keep the following points in mind: Braces ({ }) are put around the members of a set. No other symbols are used for this purpose. A set may contain only one sentence, in which case the braces before* $\cup$ *may be omitted. A set may also be empty, in which case it can be designated by* Ø, *by* { }, *or by nothing at all.*
★a. $\{(A \rightarrow B), A\} \cup \sim B$
b. $\{(A \rightarrow B), (B \rightarrow C)\} \cup \sim(A \rightarrow C)$
★c. $A \cup \sim A$
d. $A \cup A$
★e. $\{(A \rightarrow B), \sim B\} \cup \sim\sim A$
f. $\sim B \cup \sim B$

2.2 Conventions

The vocabulary of SL contains just two punctuation marks, and the third syntactic rule requires the use of parentheses with each binary connective. The ensuing forest of punctuation marks can make sentences hard to read. To improve the appearance of sentences, two conventions are adopted.

(b) Brackets may be substituted for parentheses.
(op) Outermost parentheses may be coloured out (they are still there).

These are conventions in the sense that they are not officially included in the vocabulary or allowed by the syntax of the language. In strictness, brackets should be understood to really be poorly drawn parentheses, and outer parentheses should be understood to really be there, written in ink too pale to see.

(op) only allows colouring out outer parentheses if they really are outer. To be outermost the left parenthesis must be the first vocabulary element of the sentence. In accord with this definition

2.7 $\sim(A \rightarrow A)$

has no outermost parentheses.

Outermost parentheses may only be coloured out if they remain outer. If a further vocabulary element is added to either the front or the back of a sentence, the outermost parentheses must be coloured back in before the addition occurs.

In the chapters that follow, frequent mention is made of the negation, ~P, of a sentence, P, and it is often necessary to construct negations.

> The *negation*, $\sim P$, of a sentence, P, is constructed by applying ($\sim$) to P.
>
> *Caveat:* If (op) has been applied to P, the outer punctuation must be coloured back in before applying ($\sim$).

In accord with the caveat, the negation of $A \vee B$ is $\sim(A \vee B)$. $\sim A \vee B$ cannot be constructed by negating $A \vee B$. It must be constructed by disjoining $\sim A$ and B.

$\sim A \vee B$ and $\sim(A \vee B)$ are not the same. Sentences are not the same unless they consist of the same vocabulary elements presented in the same order. Parentheses are vocabulary elements. They are officially present even when not coloured in. Parentheses serve to define the "scope" or extension of the associated binary connective. In $\sim(A \vee B)$ the leftward scope of "$\vee$" is A. In $\sim A \vee B$, otherwise written as $(\sim A \vee B)$, the leftward scope of $\vee$ is $\sim A$. This is a significant syntactic difference. It is a separate question whether there is also a difference in meaning. (In this case, there is.) For now, the only question is whether sentences are the same or different, and that question is decided by the appearance, not by the meaning.

> Two *sentences* are not *the same* unless they consist of the same vocabulary elements, presented in the same order.
>
> ***Caveat:*** When determining whether two sentences are the same, disregard the effect of applying any informal notational conventions.

Exercise 2.5

1. *State whether the sentences in each of the following pairs are the same. Give reasons to justify your answer.*

- ⋆**a.** $A, \sim\sim A$
- b. $A \,\&\, B, B \,\&\, A$
- ⋆**c.** $A \,\&\, (B \,\&\, C), (A \,\&\, B) \,\&\, C$
- d. $A \rightarrow B, (A \rightarrow B)$
- ⋆**e.** $(A \equiv [C \vee B]), [A \equiv (C \vee B)]$
- f. $\sim(A \,\&\, B), (\sim A \,\&\, \sim B)$

2. *Construct the negation of each of the following sentences.*

- ⋆**a.** $\sim A$
- b. $\sim(A \vee B)$
- ⋆**c.** $\sim A \vee B$
- d. $A \rightarrow \perp$
- ⋆**e.** $A \rightarrow (B \rightarrow \perp)$
- f. $(A \rightarrow B) \rightarrow \perp$

2.3 Syntactic Demonstrations and Trees

An expression can be demonstrated to be a sentence by showing how it is built in accordance with the formation rules. For example, $\sim A \vee B$ is demonstrated to be a sentence as follows:

> Since A is a sentence by (SL), $\sim A$ is a sentence by ($\sim$). B is also a sentence by (SL), so since it has just been established that $\sim A$ is also a sentence, it follows by (bc) that $(\sim A \vee B)$ is a sentence. So, by (op), $\sim A \vee B$ is a sentence.

This demonstration is given in the conversational style of everyday discourse. Another style of demonstration says the same things, but numbers the different assertions and separates justifications from assertions.

1. A and B are sentences (by (SL)).
2. $\sim A$ is a sentence (from line 1 by ($\sim$)).

3. (~A ∨ B) is a sentence (from lines 2 and 1 by (bc)).
4. ~A ∨ B is a sentence (from line 3 by (op)).

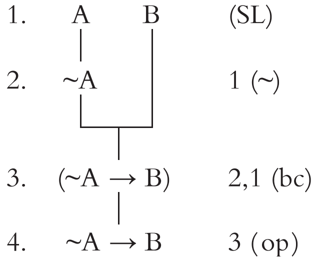
Numbering assertions is useful for referring to what was established at earlier points. As demonstrations grow longer, it can become difficult for the reader to remember all their parts. This makes it useful to have an indexing system to allow the reader to quickly look up what was established earlier.

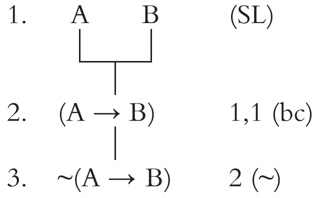
To take another example, the entirely different expression, ~(A ∨ B), is demonstrated to be a sentence by an appeal to a different sequence of steps. Since it is tedious to write "is a sentence" and "from … by" on every line, and since after seeing one demonstration written in this way everyone should understand how it goes, an abbreviated format may be adopted.

1. A (SL)
2. B (SL)
3. (A ∨ B) 1,2 (bc)
4. ~(A ∨ B) 3 (~)

At line 3 (bc) is applied to A and B rather than (~) to A, setting the course for building ~(A ∨ B) rather than ~A ∨ B.

Another way of demonstrating that an expression is a sentence is by means of syntactic trees, which diagram how the parts are put together, step by step, to form the sentence. Here are syntactic trees for the two sentences discussed previously.





Whereas living trees grow from the bottom up, these syntactic trees are drawn from the top down. There is one branch at the top of the tree for each occurrence of ⊥ or a sentence letter, taken in order from left to right as they appear in the sentence. If ⊥ or a sentence letter occurs more than once in a sentence, they are listed more than once on the first line. The first line of the tree is always justified by (SL). If (op) is applied, it may only be applied on the last line of the tree or demonstration. The intermediate lines depict how the sentence is built from its parts. Where there are differences between sentences, their trees look different.

The trees above are simple. When sentences get more complex, it can be unclear how to proceed with the tree, and there can be different ways of doing so. If it is not obvious how to proceed, it can help to box off the parts of the sentence in the following order:[1]

1. Draw a box around each sentence letter and each ⊥. This identifies each application of (SL).
2. Draw the box defined by each pair of corresponding punctuation marks. This identifies applications of (bc). Where the corresponding punctuation marks are brackets it also identifies applications of (b).
3. If a tilde is followed by a box, draw the box that contains that tilde and the box that follows it. Repeat as necessary if this creates a new case of a tilde being followed by a box. This identifies each application of (~).
4. If the previous steps have not put the whole sentence in a box, draw the box that contains the whole sentence. This identifies an application of (op).

For example, the sentence,

$$\sim[(\sim A \rightarrow A) \equiv \sim(A \,\&\, \sim\bot)] \vee (\sim A \rightarrow A) \tag{2.8}$$

can be boxed off in the following stages. Step 1 produces six boxes:

$$\sim[(\sim\underset{1}{\boxed{A}}\to\underset{2}{\boxed{A}})\equiv\sim(\underset{3}{\boxed{A}}\&\sim\underset{4}{\boxed{\bot}})]\vee(\sim\underset{5}{\boxed{A}}\to\underset{6}{\boxed{A}})$$

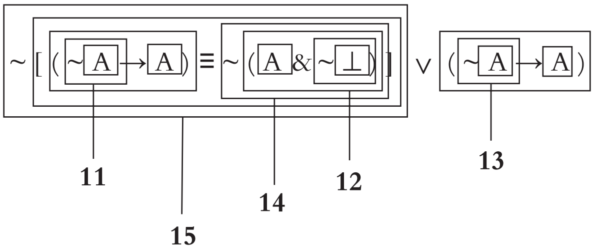
Step 2 produces four more:

$$\sim[\underset{\mathbf{7}}{\boxed{(\sim\boxed{A}\to\boxed{A})}}\equiv\sim\underset{\mathbf{8}}{\boxed{(\boxed{A}\&\sim\boxed{\bot})}}]\vee\underset{\mathbf{9}}{\boxed{(\sim\boxed{A}\to\boxed{A})}}$$

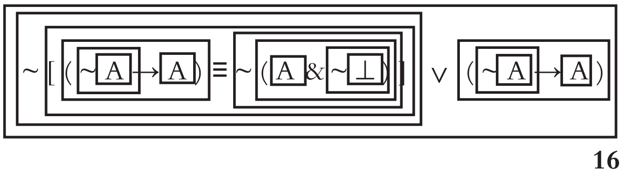
and

$$\sim[\underset{\mathbf{10}}{\boxed{\boxed{(\sim\boxed{A}\to\boxed{A})}\equiv\sim\boxed{(\boxed{A}\&\sim\boxed{\bot})}}}]\vee\boxed{(\sim\boxed{A}\to\boxed{A})}$$

Step 3 produces:



Step 4 produces:



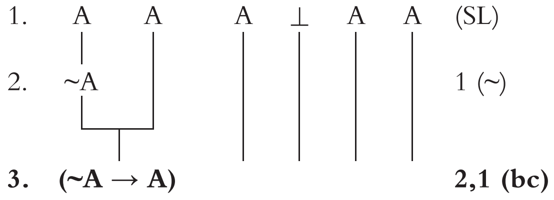
The boxes exhibit how the sentence's atomic parts are put together to make compounds that are put together to make larger compounds. (The atomic parts are ⊥ and the sentence letters.) When producing a syntactic tree, work from innermost boxes out to the boxes that contain those boxes, to the boxes that contain those boxes, and so on. For example, work from box 1 to box 11, from boxes 11 and 2 to box 7, from 4 to 12, from 3 and 12 to 8, from 8 to 14, and so on. When a larger box contains two smaller boxes separated by a binary connective, there can be a choice whether to start with the smallest boxes on the left side, or those on the right. For example, when constructing the contents of box 10, it is just as good to start by going from 4 to 12, 3 and 12 to 8, and 8 to 14 as it is to start from 1 to 11, and 11 and 2 to 7. Going from left to right incrementally generates the following tree for ~[(~A → A) ≡ ~(A & ~⊥)] ∨ (~A → A):

1. A A A ⊥ A A (SL)

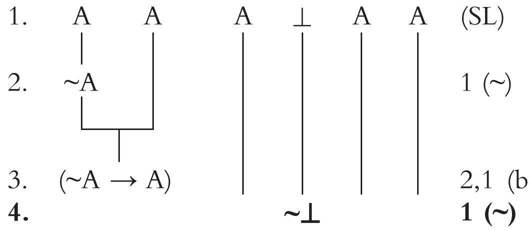
The first line of the tree always lists the atomic components as they appear from left to right. They are sentences according to (SL). Starting with the leftmost of the innermost boxes and going out means going from the sentence in box 1 to the sentence in box 11, which is done by (~).

1. A A A ⊥ A A (SL)
| | | | | |
2. ~A 1 (~)

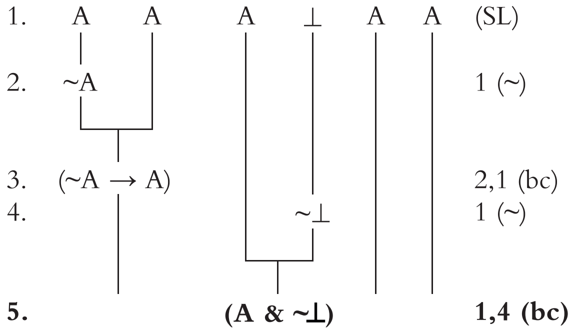
Box 11 is contained in box 7, which joins the sentence in box 11 to the one in box 2 with an arrow, so the next step is to apply (bc) to generate the contents of box 7.



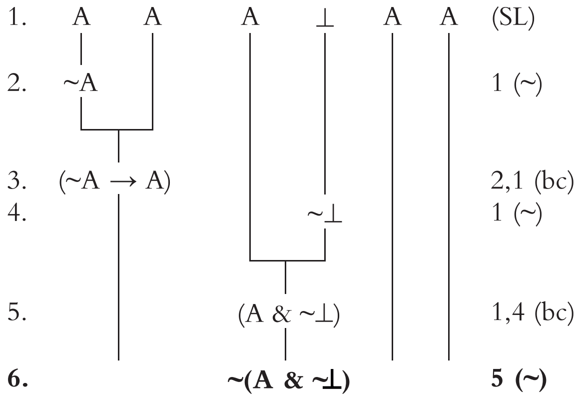
Box 7 is contained in box 10, which joins the sentence in box 7 to the one in box 14 with a triple bar. Since the sentence in box 14 is compound, the tree needs to show how it is constructed before joining it to the sentence in box 7. The leftmost of the innermost boxes in box 14 is box 3. The sentence in that box is joined to the one in box 12 with an ampersand. Since the sentence in box 12 is again compound, the tree must first show how it is constructed before joining it to box 3. This means the next step on the tree is to start with the sentence in box 4. Stepping up from it to what is contained in box 12 means applying (~):



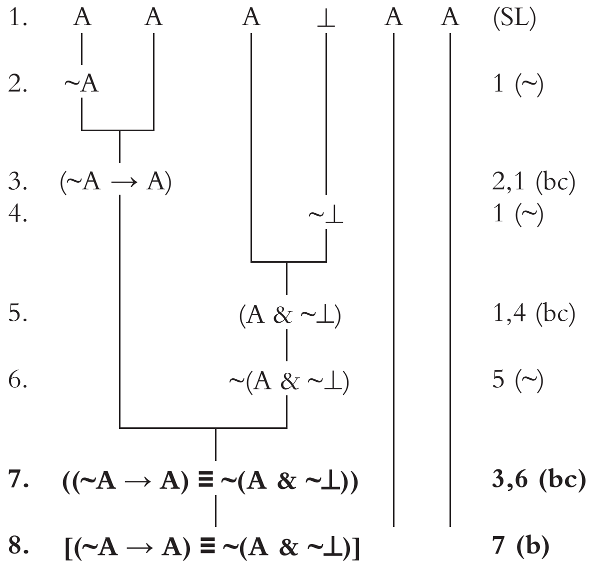
Now (bc) can be applied to the sentences in boxes 3 and 12 to generate the sentence in box 8:



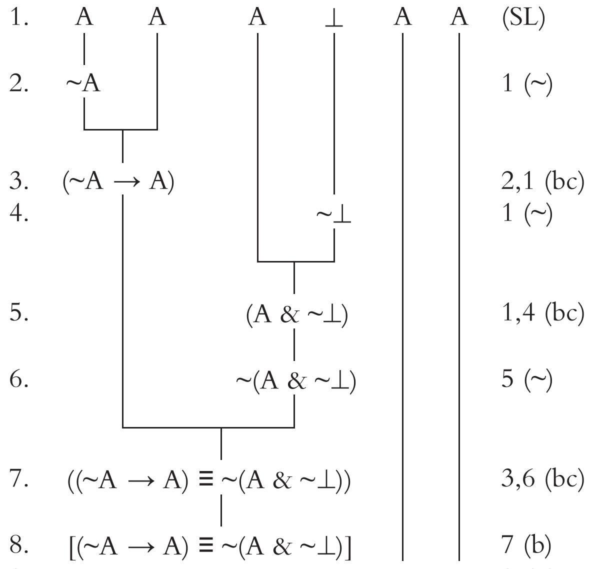
Now it is possible to go from the sentence in box 8 to the one in box 14.



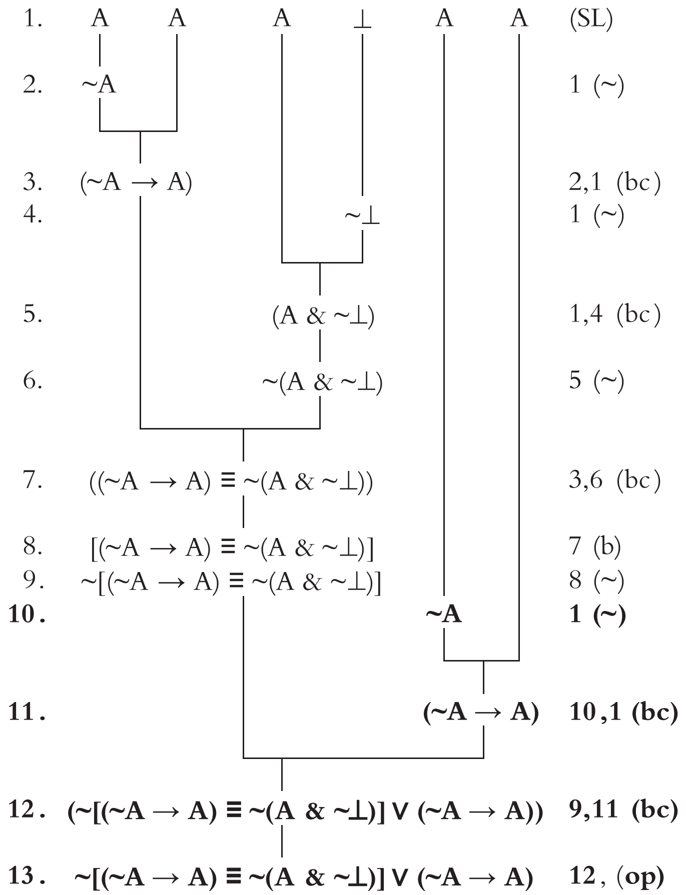
And now it is possible to go from the sentences in boxes 7 and 14 to the one in box 10. When the sentences in boxes 7 and 14 are conjoined, it must be by applying (bc), which always introduces parentheses. However, the sentence being constructed contains brackets. This necessitates a subsequent application of (b) to convert the parentheses to brackets.



Now it is possible to go up to box 15.



Box 15 is contained in box 16, which joins the sentence in box 15 to the one in box 9 with a wedge. Since the sentence in box 9 is again compound, it must first be built up from its components. Once this has been done over lines 10 and 11 applying (bc) and (op) completes the tree.



Because there are only five formation rules (including (op) and (b)), and their application is obvious, it is permissible to omit the line numbers and justifications and just draw the tree.

However, it is not permissible to omit the applications of (op) and (b). (bc) does not add brackets, and it never fails to add parentheses, so it would not be permissible to skip line 7 above and declare line 8 to be obtained by (bc), or to skip line 12 and declare line 13 to be obtained by (bc).

(op) may only be applied on the last line. It is used to remove outer punctuation and no punctuation has the status of being outer before the entire sentence has been constructed.

With practice, it becomes unnecessary to draw boxes to determine how to construct a tree. The punctuation marks and the positions of the tildes by themselves define where the boxes should go, making it possible to see how a sentence falls into parts and sub-parts without needing to draw boxes.

Exercise 2.6

Draw syntactic trees to prove that the following are sentences, being careful to apply the conventions when called for. Line numbers and justifications may be omitted.

***a.** ⊥ → [A & ~A]
b. ~~~B
c. (A ≡ ~A) ∨ ~B
d. ⊥ ≡ (~A ∨ ~B)
e. ~[~(A & B) & C]
***f.** ~([~A → B] ∨ [C & (G ∨ ⊥)]) ≡ ~G
g. [(⊥ → ⊥) → [⊥ → (A & ~A)]]
***h.** [A → (C ∨ ~C)] & [~A → (C ∨ ~C)]
i. ~(⊥ & A) ≡ (~⊥ ∨ ~A)

2.4 Scope; Main Connective and Immediate Components; Named Forms

The concepts of the scope or extension of a connective and of the main connective of a sentence are foundational for many other syntactic concepts. Intuitively, the scope of a connective is the sentences that the connective connects to (or together). On the line of a syntactic tree where a connective is added to a sentence, the scope of the connective is the prior sentence or sentences the new connective is added to. The main connective of a sentence is the last connective to be added in the process of generating the sentence. It is the only connective to have scope over the whole sentence. More rigorously, scope is defined as follows:

- when ~ is followed by an atomic sentence (⊥ or a sentence letter), its scope is that atomic sentence
- when ~ is followed by an opening punctuation mark, its scope is everything from that mark up to the corresponding closing punctuation mark, inclusive of those marks
- when ~ is followed by a tilde, its scope is the following tilde and whatever falls within the scope of that following tilde
- the scope of a binary connective (&, ∨, →, or ≡) begins and ends with the punctuation marks added to the sentence in conjunction with that connective, excluding that connective, but including its punctuation marks

---

Exercise 2.7

1. *Number the tildes in each of the following sentences from left to right. Beside the number for each tilde, write down the sentence that falls within that tilde's scope.*

★**a.** ~⊥
b. ~~⊥
★**c.** ~⊥ ∨ B
d. ~(⊥ ∨ B)
★**e.** ~(~⊥ ∨ ~B)
f. ~~~A
★**g.** ~(A & B) → ⊥
h. ~C ∨ A
★**i.** ~(C → ~A)
j. ~[~(~A & B) & C]

2. *Number the binary connectives in each of the following sentences from left to right. Beside the number for each binary connective, write down the sentence that falls within the leftward scope of that binary connective, followed by the sentence that falls within the rightward scope of that binary connective.*

★**a.** ~~A ≡ B
b. ~(~A ≡ B)
★**c.** A → (B → A)
d. (A → B) → A
★**e.** (A & B) ∨ (B ≡ C)
f. [(A ∨ B) → C] & G
★**g.** A ≡ [(A & H) ∨ K]
h. B ∨ [A ∨ (C ∨ G)]

The main connective of a sentence is rigorously defined as a connective with a scope that ranges over all the other parts of the sentence. Some sentences, those comprised of a single sentence letter or ⊥, have no main connective. The rest can only have one. A demonstration that no sentence can have more than one main connective can be found in chapter A-1.1.

The immediate components of a sentence are the sentences that remain after the main connective and its associated punctuation, if any, have been removed from the sentence. Since sentences can only have one main connective, and the connectives are only unary or binary, and the binary connectives conjoin two sentences whereas the unary attach to one, sentences can have at most two immediate components and as few as none (none if they have no main connective).

Each sentence is considered a component of itself, as are its immediate components (if any), and the immediate components of anything previously identified as a component. For example, the components of ~[(A → ~B) & C] are:

- ~[(A → ~B) & C]
- (A → ~B) & C
- A → ~B
- C
- A
- ~B
- B

Sentences and their immediate components are given special names depending on what their main connective is. The names hint at the meaning of the connectives. Those hints should be disregarded for now.

Named Forms

A sentence that has no main connective is an *atomic sentence*. Atomic sentences have no immediate components. In particular, ⊥ is an atomic sentence, as is each sentence letter.

A sentence with ~ as its main connective is a *negation*. Negations have the form ~P, where P is the immediate component of the negation. P is called the *nullation* of ~P.

The atomic sentences and the negations of atomic sentences are *literals*. A negation is not a literal unless its nullation is atomic. So ~~A is not a literal, though ~A and A are.

A sentence with & as its main connective is a *conjunction*. Conjunctions have the form P & Q, and the immediate components, P and Q, are called *conjuncts*.

A sentence with ∨ as its main connective is a *disjunction*. Disjunctions have the form P ∨ Q, and the immediate components, P and Q, are called *disjuncts*.

A sentence with → as its main connective is a *conditional*. Conditionals have the form P → Q. The left immediate component is called the *antecedent* and the right immediate component is called the *consequent*.

A sentence with ≡ as its main connective is a *biconditional*. Biconditionals have the form P ≡ Q.

Arbitrarily more specific forms can be described using combinations of these names: conditionals with negated antecedents, conjunctions with a first conjunct that is a negated conditional and a second conjunct that is a disjunction of a biconditional and a negation, and so on.

Alternatively, more specific forms can be pictured using metavariables for the unanalysed components. A negated disjunction has the form ~(P ∨ Q), a disjunction with a negated first disjunct has the importantly different form ~P ∨ Q, and a conjunction with a first conjunct that is a negated conditional and a second conjunct that is a disjunction of a biconditional and a negation has the form ~(P → Q) & [(R ≡ P_1) ∨ ~P_2].

The ability to *parse* a sentence is the ability to identify its main connective, its immediate component(s), the main connective of each immediate component, the immediate component(s) of each immediate component and so on up the syntactic tree to atomic components.

Exercise 2.8

State whether each of the following sentences is a negation, conjunction, disjunction, conditional, or biconditional. Then identify its immediate component or components and say whether that immediate component is atomic or is a negation, conjunction, disjunction, conditional or biconditional.

★a. ~A ∨ B
b. ~(A ∨ B)
★c. A ∨ ~(B ≡ A)
d. ~(A ∨ B) ≡ A
★e. K ∨ (~[G ∨ ~(B ≡ K)] → A)
f. K → ~([G ∨ ~(B ≡ K)] → A)
★g. (K ∨ G) ∨ [~B ≡ (K → A)]
h. ~[(K ∨ G) → [~B ≡ (K → A)]]
★i. [~(A & B) → (B ≡ (H &C))] & ~⊥
j. [(A ∨ ~A) & ⊥] ≡ ~(C ≡ A)

2.5 Formal Properties

Sameness
No expression or sentence of SL is *the same* as any other unless it consists of the same vocabulary elements, placed in the same order.

Two expressions or sentences are *distinct* if and only if they are not the same.

Caveat: The effects of applying informal notational conventions are to be disregarded when determining whether two expressions or two sentences are the same.

Negation
The *negation*, ~P, of a sentence, P, is constructed by applying (~) to P. If (op) has been applied to P, P's outer punctuation must be replaced before applying (~).

Opposition
Two sentences are *opposites* if and only if one of them is the negation of the other.

Formal Contradiction
A *formal contradiction* is either two sentences, one of the form P, the other of the form ~P, or the sentence ⊥, or any sentence of the form (P & ~P), or any sentence of the form (~P & P).

Converse, Inverse, and Contrapositive
Given a conditional sentence, P → Q,
the *converse* of P → Q is Q → P
the *inverse* of P → Q is ~P → ~Q
the *contrapositive* of P → Q is ~Q → ~P.

Exercise 2.9

1. *State whether the sentences in each of the following pairs are opposites. Give reasons to justify your answer. If the sentences are not opposites, identify the true opposite of each.*
★a. ~A, ~~A
b. A, ~~~A
★c. A & B, ~A & ~B
d. ~(A ∨ B), ~~(A ∨ B)
★e. ~(A ∨ B), [A ∨ B]
f. A → (B ∨ C), ~A → (B ∨ C)

2. *Construct the converse, inverse, and contrapositive of each of the following conditionals.*

- ⋆**a.** A → (B & C)
- b. ~A → ⊥
- ⋆**c.** ~⊥ → ~~⊥
- d. A → (B → C)
- ⋆**e.** (⊥ → ~⊥) → ~~⊥
- f. ~(A → B) *(trick question)*

Note

1 Not everyone has difficulty seeing how sentences are constructed. They should skip to the concluding four paragraphs of this section.

3 Semantics

Contents

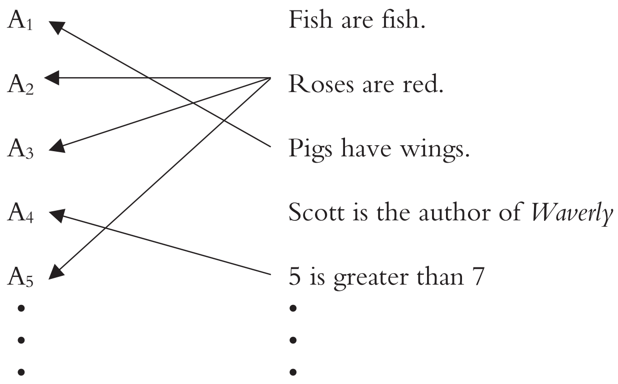
3.1 Semantics for ⊥ and the Sentence Letters

Whereas the syntax for SL specifies how the vocabulary elements can be arranged, the semantics specifies the meaning of the vocabulary elements and of their various arrangements.

The sentence letters of SL are used as variables for sentences of English or other natural languages. As such, their meaning is not fixed, but varies from case to case. A case might be an example, an exercise question, or a discussion in a textbook section.

Consider all the sentences of English to be lined up in a column. In a column to their left are all the sentence letters of SL. Now imagine each sentence letter as the terminus of exactly one arrow originating from an English sentence. Each sentence letter must be pointed to by an arrow. No sentence letter can be pointed to by more than one arrow. But the same English sentences can send arrows to multiple different sentence letters. And some English sentences can send no arrows.

DOI: 10.4324/9781003026532-4



This scheme illustrates the central semantic notion of an interpretation (hereafter often designated with the symbol, "I"). An interpretation is one way of interpreting or assigning sentences of a natural language to sentence letters. There are infinitely many others. Imagine all the different ways an arrow could be drawn to each of the sentence letters in the left column from any sentence in the right column. Each way of drawing exactly one arrow to each sentence letter is an interpretation.

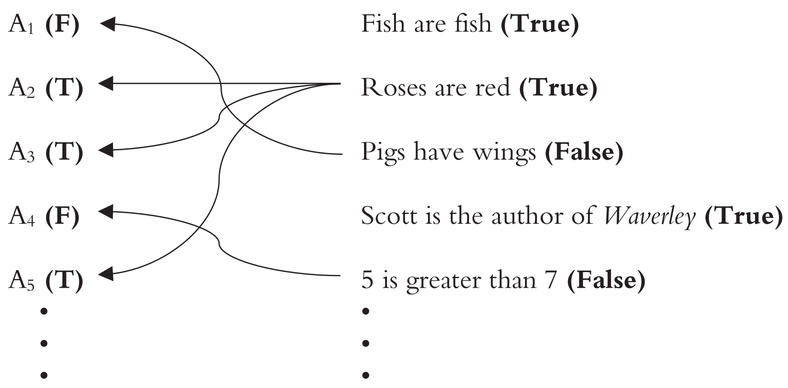
Another way to put this point is to say that interpretations are one-place functions in the sense discussed at the close of chapter 1. Each sentence letter is an "argument" the function assigns a value (an English sentence) to. The sentence letters are the things up for valuation or the "arguments." The sentences of English are the "values" that the function assigns to them. Any one interpretation is any one way of assigning exactly one value (English sentence) to each argument (sentence letter).[1]

While the sentence letters can be interpreted in infinitely many ways, there are two constraints on any one interpretation:

- Logic does not tolerate ambiguity. No sentence letter can have two different meanings on any one interpretation. This means no interpretation can assign two different sentences to a sentence letter. It does not hold the other way. There is nothing wrong with assigning the same English sentence to two different sentence letters. That just means that different sentence letters are names or symbols for the same sentence. The meaning of each sentence letter is still unambiguous.
- No sentence letter can be meaningless. Each must have exactly one sentence assigned to it. Again, it does not hold the other way. Some sentences can go unsymbolized. In the extreme, one sentence might be assigned to all the sentence letters, leaving all the remaining sentences unsymbolized.

Granting that these two constraints have been met, a question arises about values. There are many values that might be ascribed to sentences: true, false, both true and false, neither true nor false, indeterminate, evident, nonevident, possibly true, true to degree *x*, obeyed, disobeyed, kept, broken, answered, unanswered, answerable, unanswerable, and the list goes on. A simple place to start is with just two values, true and false. (Other values will be introduced after learning how things work with just these two.) Confining consideration to just those English sentences that can bear exactly one of these two values means placing a further (temporary) restriction on interpretations. They may only assign those English sentences that bear exactly one of the two values to sentence letters. Paradigmatically, these are assertions. But the range of English sentences up for consideration may be broadened to include sentences that do other things in addition to making an assertion. Someone who makes a promise might be said to perform the act of bringing an obligation into being. But part of making a promise is declaring that an event within the agent's control will occur at a certain time provided the right circumstances have been met, and this is an assertion about the future course of events that can prove to be true or false. Someone who issues a command or asks a question simultaneously makes a (necessarily true) autobiographical report ("I order you to …"; "I would like to ask you whether …"). That report can be picked up and anonymously reissued by others ("It is required that …"; "It is questionable whether …"), and these restatements may be true or false depending on how accurately they represent what was originally commanded or asked.

By assigning sentences that make an assertion to sentence letters, interpretations invest those sentence letters with values. A sentence letter acquires the value true if the sentence assigned to it is true, and false if the sentence is false. Often, these values are symbolized as T and F. However, it is not uncommon to use other symbols (1 and 0 are popular but are needed for other purposes in this textbook), and to associate those symbols with other pairs of opposed values, such as "switched on" and "switched off." For now, T means "true" and F means "false," but in later chapters these symbols will be extended to stand for other values and taken to be determined by other values.



For sentence letters, truth is relative. There are infinitely many ways of drawing a single arrow to each sentence letter from one of the sentences of a natural language like English. Each corresponds to a different interpretation. The value of the sentence letters changes depending on how they are interpreted. A sentence letter is always true or false *on an interpretation*. No sentence letter is absolutely true or false.

The discussion so far has not mentioned ⊥, and ⊥ has not appeared on any of the columns of sentences and sentence letters displayed above. ⊥ is not a variable. It is a part of the logical vocabulary of SL, like the connectives. This makes it an exception. It has a meaning that is not open to interpretation. The meaning of ⊥ is F.[2]

When speaking of the values of sentences it is common to use expressions like "A is true on I," "B is true on J," "C is false on I," where A, B, and C are sentence letters and I and J are interpretations. In writing, it saves time and effort to employ a more abbreviated form: I(A) is T, J(B) is T, I(C) is F.

It is also common to display interpretations on a table.

	A	B	C	G	H	K	A_1	A_2	…
I	T	T	F	F	T	T	F	T	…

On this table the sentence letters are listed in alphanumeric order on the top row and the values I assigns to each sentence letter are listed on the bottom row. ⊥ is not listed as it is not up for interpretation.

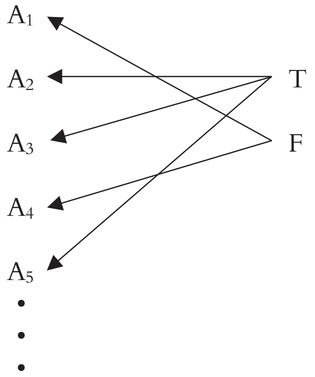
When two interpretations, I and J, are under consideration they can be compared on the same table.

	A	B	C	G	H	K	A_1	A_2	…
I	T	F	F	F	T	T	F	T	…
J	T	T	F	F	T	T	F	T	…

The table shows that J is like I but for assigning T to B, and perhaps in other ways as well that do not make it onto the displayed portion of the table.

The difference in the assignments that I and J make to B would be explained by the fact that they assign different sentences of English to B. Perhaps I assigns "snow is green" to B and J assigns "grass is green" to it.

In logic, it is often unimportant what sentence is assigned to a sentence letter. It only matters whether it is assigned to a true sentence or a false one. It is irksome to have to make the connection between sentence letters and values by appeal to sentences. The truth or falsity of sentences is often dependent on the facts or the context in which the sentence is uttered, and that requires that these facts and contexts be known or ascertained. To get around this, logicians often skip over identifying which sentence is assigned to which sentence letter. Instead, they just consider whether a sentence letter has one of the true sentences or one of the false ones assigned to it. This is tantamount to treating the sentence letters as if they were each assigned one of the values, rather than assigned sentences with these values.



This gives rise to two ways of understanding an interpretation: as an assignment of sentences of a natural language to sentence letters of SL that results in an assignment of values to sentence letters of SL, or as a direct assignment of values to sentence letters of SL, ignoring the intermediate assignment of sentences of English. An interpretation that works in the second of these ways is called a valuation.

> A *valuation* is an assignment of values to the nonlogical vocabulary elements of a formal language.
>
> A *valuation for SL* is an assignment of exactly one of T and F to each sentence letter. Notation: I(P) is T or I(P) is F, where P is a sentence letter.

A valuation for SL is a function on the domain of sentence letters. It specifies a co-domain, V, of values, {T,F}. It takes each sentence letter as an argument and returns exactly one of T or F as value. There are infinitely many different ways of assigning exactly one of T and F to each sentence letter. Each of these ways is a different valuation and so a different interpretation. Different interpretations return different values in the case of at least one sentence letter.

Exercise 3.1

1. *Identify what value the following interpretations assign to the specified sentence letter.*

- **★a.** I(A) is "The Sun is an astronomical object."
- b. I(B) is "The Eiffel Tower is an astronomical object."
- **★c.** I(C) is "The Eiffel Tower is a terrestrial object."
- d. I(G) is "The Sun is the Eiffel Tower."
- **★e.** I(H) is "The Sun is the Sun."
- f. I(K) is "The Eiffel Tower is not the Eiffel Tower."
- **★g.** Is it possible for the same interpretation to make all of the assignments described in (a)–(f)? Why or why not?

2. *Identify a sentence that* I *might assign to the indicated sentence letter or sentence letters in order to assign the identified value. Draw on sentences that are generally well known to be true or false, not those that are controversial or that others would have to research to learn about. In cases where questions identify assignments to two or more sentence letters, consider whether one sentence could do the job for more than one sentence letter and make identifications accordingly. Consult the answered questions for further guidance.*

- **★a.** I(A) is F
- b. I(B) is T
- **★c.** I(A) is T and I(C) is T
- d. I(A) is F and I(C) is T
- **★e.** I(A) is T, I(B) is T, I(C) is F
- f. I(A) is F, I(B) is T, I(C) is T

Logicians are not concerned which of the infinitely many interpretations is the correct one. It is standard practice to consider all of them.

This is not always as big a job as it might seem. When considering one sentence letter, A, the infinitely many interpretations reduce to two: the ones that assign T to A and the ones that assign F to A. After all, if the interest is just in A, all that needs to be considered is whether an interpretation assigns it T or F, regardless of how values are assigned to all other sentence letters.

Similarly, when considering two sentence letters, A and B, the infinitely many interpretations reduce to four, the ones that assign T to both A and B, the ones that assign T to A but F to B, the ones that assign F to A but T to B, and the ones that assign F to both.

When considering three sentence letters, there are 8 interpretations; four, 16; five, 32; and so on, though for practical purposes more than five will rarely be considered. The number of different interpretations is 2^n, where n is the number of different sentence letters under consideration.

When the number of sentence letters is not too great, there is a standard form for listing interpretations. The interpretations are presented on a table. On the top row of the table, the sentence letters are listed in alphanumeric order (A–K first, followed by the subscripted A's). Below this row is the appropriate number of rows for that number of sentence letters (2^n where n is the number of atomic sentences). In the rightmost column, the one for the last sentence letter in the alphanumeric order, T and F alternate for the appropriate number of rows. In the next column to the left, two T's alternate with two F's, then four with four, eight with eight, and so on out to the first sentence letter in the alphanumeric order.

The alteration is based on the following considerations: For a single sentence letter, A, there are just two kinds of interpretations: those that assign T and those that assign F.

	A
I_T	T
I_F	F

When two sentence letters, A and B, are under consideration, each of these cases splits into two. The interpretations that assign T to A divide into those that assign T to B and those that assign F to B, and likewise for the interpretations that assign F to A.

	A	B
I_{T1}	T	T
I_{T2}	T	F
I_{F1}	F	T
I_{F2}	F	F

When three sentence letters, A, B, and C, are under consideration, each of these four cases again splits into two. The interpretations that assign T to A and B divide into those that assign T to C and those that assign F to C, and likewise for the other three groups of interpretations, producing 4 × 2 or 8 groups:

	A	B	C
$I_{T1\text{-}1}$	T	T	T
$I_{T1\text{-}2}$	T	T	F
$I_{T2\text{-}1}$	T	F	T
$I_{T2\text{-}2}$	T	F	F
$I_{F1\text{-}1}$	F	T	T
$I_{F1\text{-}2}$	F	T	F
$I_{F2\text{-}1}$	F	F	T
$I_{F2\text{-}2}$	F	F	F

Each time a further sentence letter is added to the table, each of the previously listed types of interpretation splits into two groups, thereby continually doubling the number of listed interpretations.

Exercise 3.2

1. *Tabulate the possible interpretations for the following collections of sentence letters in standard form.*

- **★a.** A, ⊥
- b. C, A
- **★c.** A_1, B
- d. G, A, C
- **★e.** C, A, A_3, B
- f. A_1, A_4, G, A_{10}, A_2

2. *State what is wrong with the following tables of interpretations (each has something wrong with it).*

★a.

	A	⊥
I_1	T	F
I_2	T	F
I_3	F	F
I_4	F	F

b.

	A	B
I_1	T	T
I_2	F	T
I_3	T	F
I_4	F	F

★c.

	A	B	C
I_1	T	T	T
I_2	T	T	F
I_3	T	F	T
I_4	F	F	F
I_5	F	F	T
I_6	F	T	F
I_7	T	F	F
I_8	F	T	T

d.

	A_1	B	C
I_1	T	T	T
I_2	T	T	F
I_3	T	F	T
I_4	T	F	F
I_5	F	T	T
I_6	F	T	F
I_7	F	F	T
I_8	F	F	F

★e.

	C	G
I_1	T	T
I_2	F	T
I_3	T	F
I_4	F	F
I_5	T	T
I_6	F	T
I_7	T	F
I_8	F	F

f.

	A	B	C
I_1	T	T	T
I_2	T	F	F
I_3	T	T	T
I_4	F	F	F
I_5	F	T	T
I_6	F	F	F

3.2 Semantics for the Connectives

The connectives of SL build compound sentences that have a value that is completely determined by the values of their immediate components. Each connective does this in a different way. The meaning of each connective is given by the way it does this. ⊥ can be treated as if it were a connective.

⊥	builds an atomic sentence that is false on any interpretation
~	builds a compound sentence that has the opposite value of its immediate component
&	builds a compound sentence that is true if and only if both immediate components are true and otherwise is false
∨	builds a compound sentence that is true if and only if at least one immediate component is true and otherwise is false
→	builds a compound sentence that is false if and only if its antecedent is true but its consequent is false and otherwise is true
≡	builds a compound sentence that is true if and only if its immediate components have the same value and otherwise is false

Keeping in mind that truth is truth on an interpretation, the connectives are symbols for rules that an interpretation follows in assigning values to compound sentences, depending on what connective is used to build the compound. These rules can be called valuation rules.

Valuation Rules

(⊥): I(⊥) is F
(~): I(~P) is T if and only if I(P) is F; otherwise I(~P) is F
(&): I(P & Q) is T if and only if I(P) and I(Q) are T; otherwise, I(P & Q) is F
(∨): I(P ∨ Q) is T if and only if at least one of I(P) and I(Q) is T; otherwise I(P ∨ Q) is F
(→): I(P → Q) is F if and only if I(P) is T and I(Q) is F; otherwise I(P → Q) is T
(≡): I(P ≡ Q) is T if and only if I(P) is the same as I(Q); otherwise I(P ≡ Q) is F

This statement of the valuation rules is optimized for both use and concision. Though the format is different from the informal statement given earlier, the two versions come to the same thing.

(~) was used in chapter 2 as a label for a syntactic formation rule. It is reused here as a label for a semantic rule. There is no danger of confusing the two.

The valuation rules might be more expansively presented by unpacking the phrases, "if and only if" and "otherwise."

"Otherwise" means "in all other cases." At this point only two cases are under consideration, so specifying the "other" cases gives rise to a statement of both the conditions under which I assigns a T to a compound and the conditions under which it assigns an F.

(T~) I(~P) is T if and only if I(P) is F	**(F~)** I(~P) is F if and only if I(P) is T
(T&) I(P & Q) is T if and only if I(P) and I(Q) are T	**(F&)** I(P & Q) is F if and only if at least one of I(P) and I(Q) is F
(T∨) I(P ∨ Q) is T if and only if at least one of I(P) and I(Q) is T	**(F∨)** I(P ∨ Q) is F if and only if I(P) and I(Q) are F
(T→) I(P → Q) is T if and only if at least one of the following: (i) I(P) is F; (ii) I(Q) is T	**(F→)** I(P → Q) is F if and only if I(P) is T and I(Q) is F
(T≡) I(P ≡ Q) is T if and only if I(P) are I(Q) are the same	**(F≡)** I(P ≡ Q) is F if and only if I(P) and I(Q) are different

These rules can be further expanded by unpacking their "if and only if" clauses. An English sentence of the form, "A if B" asserts that B is sufficient for A. B's being the case is all that is needed for A to be the case.

An English sentence of the form, "A only if B" says something else: that B is necessary for A. B's being the case is one thing (not necessarily the only thing) that is required for A to be the case.

While "A if B" says that B is all that is needed for A, it does not rule out other ways of getting A than by having B. So, it does not say that B is necessary for A. By contrast, while "A only if B" says that B is one thing that is required for A, it does not say that B is the only thing that is required for A. So, it does not say that B is sufficient for A.

While "A only if B" does not mean the same thing as "A if B," it does mean the same thing as "If A then B." "If A then B" says that A is all that is needed to get B. "A" suffices for B. So, having A means having B. "A only if B" says that having A requires having B. B is necessary for A. So, having A again means having B.

In light of these points, "A if and only if B" can be seen to assert two different things:

- "A if B," which is the same as "if B, A" or "if B then A"
- "A only if B," which is the same as "if A then B"

Using terminology that was introduced in chapter 2, "A if and only if B" asserts both "if A then B," and its converse, "if B then A." It is like a conditional that goes both ways, from left to right, so to speak (from A to B), and from right to left (from B back to A).

Because the valuation rules are stated using "if and only if," they can be read "in reverse," so to speak. They do not just specify the conditions under which I assigns a T or an F to compound sentences of different sorts. They also specify what assignments I must have earlier made to the components to be able to make the assignment it did to the compound.

Unpacking the "if and only if" clauses into an "if" clause and an "only if" clause gives the following, compound to component and component to compound statement of the rules:

Compound to Component

(for making short tables and demonstrating that there is no model)

(T~) If I(~P) is T then I(P) is F	**(F~)** If I(~P) is F then I(P) is T
(T&) If I(P & Q) is T then I(P) is T If I(P & Q) is T then I(Q) is T	**(F&)** If I(P & Q) is F then at least one of I(P) and I(Q) is F
(T∨) If I(P ∨ Q) is T then at least one of I(P) and I(Q) is T	**(F∨)** If I(P ∨ Q) is F then I(P) is F If I(P ∨ Q) is F then I(Q) is F
(T→) If I(P → Q) is T then at least one of the following: i) I(P) is F; ii) I(Q) is T	**(F→)** If I(P → Q) is F then I(P) is T If I(P → Q) is F then I(Q) is F
(T≡) If I(P ≡ Q) is T then I(P) are I(Q) are the same	**(F≡)** If I(P ≡ Q) is F then I(P) and I(Q) are different

In the case of (T&), (F∨), and (F→), one or both of the listed inferences may be drawn.

Component to Compound

(for making long tables and verifying that there is a model)

(T~) If I(P) is F then I(~P) is T	**(F~)** If I(P) is T then I(~P) is F
(T&) If both I(P) and I(Q) are T then I(P & Q) is T	**(F&)** If I(P) is F then I(P & Q) is F If I(Q) is F then I(P & Q) is F
(T∨) If I(P) is T then I(P ∨ Q) is T If I(Q) is T then I(P ∨ Q) is T	**(F∨)** If both I(P) and I(Q) are F then I(P ∨ Q) is F
(T→) If I(P) is F then I(P → Q) is T If I(Q) is T then I(P → Q) is T	**(F→)** If I(P) is T and I(Q) is F then I(P → Q) is F
(T≡) If I(P) is the same as I(Q) then I(P ≡ Q) is T	**(F≡)** If I(P) is different from I(Q) then I(P ≡ Q) is F

In the case of (F&), (T∨), and (T→), one or both of the listed inferences may be drawn.

Exercise 3.3

1. *State what, if anything, follows from each of the following by each of the valuation rules. Answers to these questions require applying the valuation rules by reasoning from values of components to values of compounds. Consult the answered questions for further illustration of how to do this.*

★**a.** $I(P)$ is T
b. $I(P)$ is F
★**c.** $I(P)$ is T and $I(Q)$ is T
d. $I(P)$ is T and $I(Q)$ is F
★**e.** $I(P)$ is F and $I(Q)$ is T
f. $I(P)$ is F and $I(Q)$ is F
★**g.** $I(P)$ is the same as $I(Q)$
h. $I(P)$ is not the same as $I(Q)$

2. *State what, if anything, follows from each of the following by a* single *application of the appropriate valuation rule. Answers to these questions require applying the valuation rules by reasoning from values of compounds to values of* immediate *components. (Do not draw conclusions for components of immediate components.) Consult the answered questions for further illustration of how to do this.*

★**a.** $I({\sim}P)$ is T
b. $I({\sim}P \vee Q)$ is F
★**c.** $I({\sim}P \,\&\, Q)$ is F
d. $I({\sim}P)$ is F
★**e.** $I(P \,\&\, {\sim}Q)$ is T
f. $I({\sim}P \,\&\, Q)$ is F
★**g.** $I({\sim}(P \,\&\, Q))$ is F
h. $I({\sim}(P \vee Q))$ is T
★**i.** $I({\sim}P \vee Q)$ is T
j. $I(P \vee {\sim}Q)$ is F
★**k.** $I(P \rightarrow (Q \vee R))$ is T
l. $I({\sim}P \rightarrow Q)$ is F
★**m.** $I((P \rightarrow Q) \equiv (Q \rightarrow P))$ is T
n. $I({\sim}(P \rightarrow Q) \equiv ({\sim}Q \rightarrow P))$ is F
★**o.** $I({\sim}[(P \rightarrow Q) \equiv ({\sim}Q \rightarrow P)])$ is T
p. $I((P \rightarrow Q) \equiv {\sim}R)$ is F

3. *State what follows from each of the following by first reasoning from the given value of the given compound to the value(s) of its component(s) and then reasoning from the given value of the sentence to the value of a negation, conjunction, disjunction, conditional, or biconditional containing that sentence as one of its immediate components. Consult the answered questions for further illustration of how to do this.*

★**a.** $I({\sim}P)$ is T
b. $I({\sim}P)$ is F
★**c.** $I(P \,\&\, Q)$ is F
d. $I(P \,\&\, Q)$ is T
★**e.** $I(P \vee Q)$ is T
f. $I(P \vee Q)$ is F
★**g.** $I(P \rightarrow Q)$ is T
h. $I(P \rightarrow Q)$ is F
★**i.** $I(P \equiv Q)$ is F
j. $I(P \equiv Q)$ is T

Over the course of this section, the valuation rules have been defined in two ways, first concisely, then expansively. There is a third way the valuation rules might be defined: as functions. The valuation rules can be considered in a very abstract way, one that does not even mention sentence letters, let alone sentences of English or other natural languages. Considered at this level of abstraction, the valuation rules are operations that assign exactly one of the values, T and F, to lists of the values T and F. ~ is a one-place function from lists of one value onto one of the values. In other words, it assigns exactly one of T and F to each list of one T or one F. Each of the binary connectives is a different two-place function. Each assigns exactly one of T and F to each list of two of the values. Each connective does this in a different way, in accord with its own valuation rule. ~ assigns F to the list <T> and T to the list <F>. & assigns T to the list <T,T> and F to the three remaining lists of two values, <T,F>, <F,T>, and <F,F>.

~(<T>) is F
~(<F>) is T

&(<T,T>) is T
&(<T,F>) is F
&(<F,T>) is F
&(<F,F>) is F

∨(<T,T>) is T
∨ (<T,F>) is T
∨ (<F,T>) is T
∨ (<F,F>) is F

→(<T,T>) is T
→(<T,F>) is F
→(<F,T>) is T
→(<F,F>) is T

≡(<T,T>) is T
≡(<T,F>) is F
≡(<F,T>) is F
≡(<F,F>) is T

While this abstract way of understanding how the valuation rules work is important, and will be brought up again in the appendix to this chapter, it will not be used for the time being. Functions go from arguments to values. In the case at hand, they go from previously made assignments to components to values that are then assigned to compounds. But in sentential logic, it is equally important to go in the other direction – that is, to think about what values the components must or might have in order to account for why the compound has the value it does. Even when going from components to compounds, the functional statement is more specific than it needs to be. In many cases, it is not necessary to know the values of both components in order to determine the value assigned to the compound. If either conjunct is false, the conjunction is false, regardless of the value of the other conjunct. If either disjunct is true, the disjunction is true regardless of the value of the other disjunct, and likewise for conditionals with a false antecedent and conditionals with a true consequent. If a biconditional is false, often all that matters is that the components have different values, not which is the one that gets the T.

Similar problems infect a further way of understanding the valuation rules. The effects of applying four of the five valuation rules can be represented on "characteristic tables." Characteristic tables combine the function tables given above, with tables of interpretations. Whereas tables of interpretations list all the ways of making assignments to the sentence letters, characteristic tables list all the ways of making assignments to immediate components. They then go on to specify the value that a valuation rule determines for the compound on each of those assignments.

According to (~) a negation receives the opposite value of its nullation. This is illustrated by the characteristic table for ~.

P	~P
T	F
F	T

According to (&) a conjunction receives a T if and only if both conjuncts get T and otherwise receives an F. This is illustrated by the characteristic table for &.

P	Q	P & Q
T	T	T
T	F	F
F	T	F
F	F	F

According to (∨) a disjunction receives a T if and only if at least one of its disjuncts gets a T and otherwise receives an F. This is illustrated by the characteristic table for ∨.

P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

And according to (≡) a biconditional receives a T if and only if its immediate components receive the same value and otherwise receives an F. This is illustrated by the characteristic table for ≡.

P	Q	P ≡ Q
T	T	T
T	F	F
F	T	F
F	F	T

Setting the different characteristic tables alongside one another provides an aerial view of how the valuation rules work. A glance shows how the assignments made by (&) differ from those made by (∨), or (≡).

P	Q	P & Q	P ∨ Q	P ≡ Q
T	T	T	T	T
T	F	F	T	F
F	T	F	T	F
F	F	F	F	T

Despite their utility for various purposes, characteristic tables are, like functional definitions of the valuation rules, underdetermined in one respect and overdetermined in another.

Using tables makes it easy to work from components to compounds: just identify the row with the given pair of component values on the table and scan across to see the value assigned to the compound in that case. But working with tables makes it harder to go in the reverse direction, since compounds can receive the same value on multiple rows, leading to multiple different conclusions about the values of the components. For example, if a conjunction is false, the characteristic table makes it false on three different rows that reveal three different pairs of values its two conjuncts could have. It seems like there are three different alternatives that have to be juggled. In fact, there is only one. A conjunction is false if and only if at least one component is false. The characteristic table does not reveal that rule. The valuation rule does.

A characteristic table for (→) has so far been neglected because it is misleading. The spatial arrangement of T's and F's on the table adds information that is not entailed by the rule. A table that does not facilitate misinterpretation would have to be incomplete.

			P → P′
row 1	T	T	T
row 2	T	F	
row 3	F	T	
row 4	F	F	T

This table puts nothing above the two left columns of values. It also makes no assignments to → on rows 2 and 3. This is not because the values are unknown or nonexistent. They are just not determined by the information that is present on the table. To fill in the blanks, it must be known which of P and P′ is the component receiving which column of values. Deciding that question and assigning T's and F's to P → P′ accordingly, and then presenting this information on a table as if it were *the* characteristic table for →, can invite the mistaken inference that the assignment is determined by which row of the table the F appears on rather than by the assignments made to P and P′ on that row. The confusion does not matter for any of the other connectives. In no other case does the order of the immediate components make any difference to the assignment to the compound.

Covering both alternatives at once can also be confusing.

	P	P′	P′ → P	P → P′
row 1	T	T	T	T
row 2	**T**	**F**	**T**	**F**
row 3	**F**	**T**	**F**	**T**
row 4	F	F	T	T

This table gives a false appearance of variety. It makes it look like the rule reverses the way it assigns T and F between rows 2 and 3. In fact, the rule always puts F in the same place, the place where the antecedent gets T and the consequent gets F. (This happens on both row 2 last column, and row 3 second last column.) The apparent variety arises only because, when P and P′ are flipped between antecedent and consequent positions, they carry their values along with them and that changes the row on which there is a true antecedent and a false consequent. The change results from a change in the place where the true component is put relative to the false component, not a change in the values of the components or a change in how (→) applies.

3.3 Semantics for Compound Sentences

The sentences of SL have two sorts of meaning: extensional meaning and intensional meaning.

The *extensional* meaning of a sentence on an interpretation, I, is its value on I.

The *intensional* meaning of a sentence is its value on each possible interpretation.

Intensional meaning is uninteresting for atomic sentences. Since an interpretation is any way of assigning exactly one of T and F to each sentence letter, and ⊥ is F on any interpretation, there could always be an interpretation on which any sentence letter is assigned a different value from ⊥ or on which any two sentence letters are assigned different values. Thus, each sentence letter is intensionally distinct from ⊥ and from each other sentence letter. This does not hold for compound sentences. Different compound sentences can have the same intensional meaning, that is, they can have the same value on each interpretation. Some can have other, noteworthy forms of intensional meaning. Extensional meaning must be considered first, however, as the range of extensional meanings determines intensional meaning.

3.3.1 Extensional Meaning

The extensional meaning of a sentence on an interpretation, I, is determined by working up from the values I assigns to its atomic components using the valuation rules. The atomic components are ⊥ and the sentence letters. The value of ⊥ is F on all interpretations. On any one interpretation, I, the value of the sentence letters is assigned by I. The value (on I) of any compound that has ⊥ or the sentence letters as its immediate components is determined by the values (on I) of those components and the rule for the compound's connective. The value (on I) of any compound that has those compounds as its immediate components is determined by their values (on I) and the rule for the main compound's connective. And so on, up to the main connective, which gives the sentence its value (on I).

When determining values on an interpretation, it is important to be able to see the architecture of the sentences of SL at a glance. This means seeing what the main connective is, what the immediate components are, what the main connective of each immediate component is, and so on down to atomic components. The ability to parse sentences of SL (to be able to identify main connectives and immediate components, and break sentences down into their component structure) is essential for further progress in the study of logic. If the parsing is unclear, drawing the architecture boxes described in chapter 2.3 should help. As a reminder, boxes are drawn by following these steps, in the order they are written:

- Draw a box around each atomic component.
- Draw the box defined by each pair of corresponding punctuation marks.
- Where there is a tilde that is followed by a box, draw the box that contains that tilde and the following box. Repeat this step as necessary when applying it puts boxes in front of further tildes.
- Draw the box containing the whole sentence, if prior steps have not already produced it.

The boxes illustrate how the sentence is compounded from its innermost parts (those in the innermost boxes) to its outermost.

Exercise 3.4

Box the parts of the following sentences following the instructions just given.

★**a.** $\sim(\sim A \vee B)$
b. $\sim\sim A \vee B$
★**c.** $\sim\sim(A \,\&\, B) \rightarrow (\sim A \equiv B)$
d. $\sim[(A \,\&\, B) \rightarrow (\sim A \equiv B)]$
★**e.** $\perp \equiv [(\sim A \,\&\, B) \vee [C \rightarrow (G \equiv H)]]$
f. $\sim[\perp \,\&\, \sim(A \,\&\, B)] \vee (\sim A \vee \sim B)$

Values are calculated by working from the innermost boxes out. The smallest boxes contain $\perp$ or sentence letters. $\perp$ is always F and the sentence letters have values that are given by I. Larger boxes contain either one immediately smaller box preceded by a tilde, or two immediately smaller boxes separated by a binary connective. The "right-to-left" (component to compound) version of the valuation rule together with the values in the immediately smaller boxes determines the value of the next box up. When the rule requires that only one of two components have a specified value and the component in the appropriate box has that value, only the value in that box need be considered when assigning a value to the larger box.

Exercise 3.5

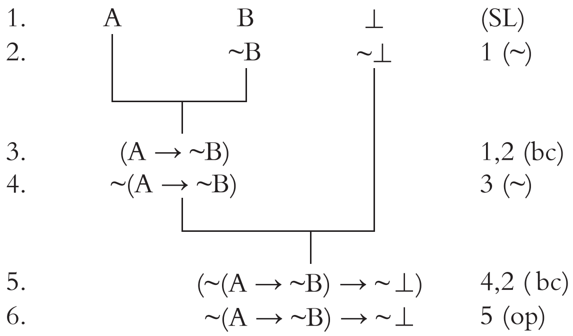
Proceeding from innermost boxes out, enter values in the boxes created for the sentences in exercise 3.4 using the interpretation that assigns T *to* A, C *and* H, *and* F *to* B *and* G.

Answers for **a, c,** and **e**

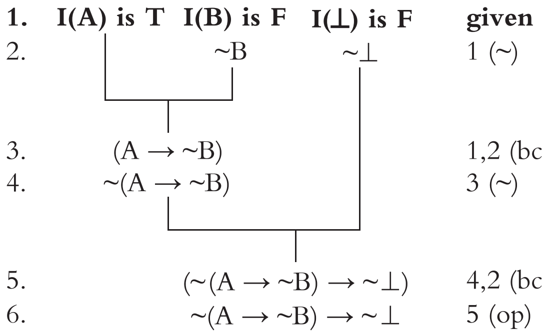
Once the ability to parse a sentence of SL has been developed, questions can be answered without drawing boxed sentences. The value of a sentence can instead be demonstrated from a given assignment to the atomic components.

For example, the value of $\sim(A \rightarrow \sim B) \rightarrow \sim\perp$ on I is determined by looking up what values I assigns to A and B. Once those values have been determined, the value of $I(\sim B)$ and $I(\sim\perp)$ can be determined. Given values for $I(A)$ and $I(\sim B)$, the value of $I(A \rightarrow \sim B)$ can be determined. (In some cases, the value of $I(A \rightarrow \sim B)$ can be determined from just one of $I(A)$ and $I(\sim B)$.) Given a value for $I(A \rightarrow \sim B)$, the value of $I(\sim(A \rightarrow \sim B))$ can be determined. And given values for $I(\sim(A \rightarrow \sim B))$ and $I(\sim\perp)$, the value of $I(\sim(A \rightarrow \sim B) \rightarrow \sim\perp)$ can be determined. (Though this is one case where both values do not need to be known for the rule to deliver a result.)

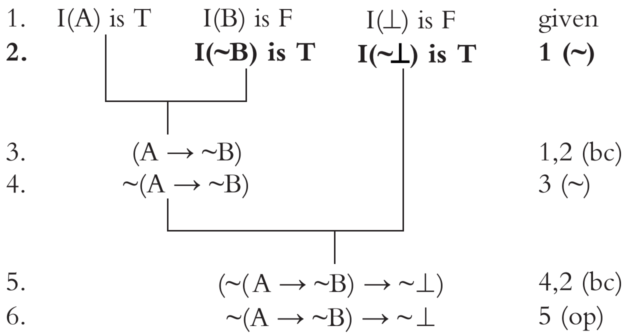
There are many equally good ways to do this. Two are discussed here: using syntactic trees, and using skeletal semantic trees. To use the method of syntactic trees, first make the syntactic tree.



Then go to the first line and convert the atomic components that appear on that line to the assignments I makes to those components. Suppose I assigns T to A and F to B. By (⊥), ⊥ gets F on any interpretation. These can be notated as the assignments that are "given."



Now go down the tree applying the valuation rules in the place of the formation rules. For example, if the syntactic rule (~) is used to add ~ to a sentence, instead apply the semantic rule (~) to calculate the value of the negation given the value previously assigned to its nullation.



If (bc) is used to conjoin two previously formed sentences with a binary connective, instead apply the semantic rule for that connective to calculate the value of the compound given the values previously assigned to its components.