Tutorial Letter 102/3/2018

Formal Logic 2

COS2661

Semester 1 & 2

School of Computing

IMPORTANT INFORMATION:

This letter will help you master the material in the prescribed book.

Define tomorrow.

UNISA university of south africa

CONTENTS

FOREWORD

Dear Student

Welcome to the second-year course in Formal Logic. We hope you will find the course interesting and challenging. Please make sure that you have read Tutorial Letter 101 thoroughly and that you are sure of all the details relevant to the sending in of assignments and the general way in which the course will run. Notice that your assignment questions, closing dates for assignments, and a study programme are also part of the contents of Tutorial Letter 101.

This Tutorial Letter 102 consists of two parts:

- Section A: Propositional Logic
- Section B: Quantifiers

The tutorial letter should be read in conjunction with your prescribed textbook because references in this tutorial letter pertain to the prescribed textbook:

> ***Language, Proof and Logic (2nd Ed) by D. Barker-Plummer, J. Barwise & J. Etchemendy, 2011. Stanford: Center for the Study of Language and Information.***

Note: In this module we cover Chapters 1 to 14 of the prescribed book, but ***not all sections of the book are prescribed. See next page***.

The purpose of Tutorial Letter 102 is to serve as supplementary material to guide and support you in your study process. We will provide you with a list of objectives and outcomes related to the study material. The outcomes should be used as a checklist when you work through the relevant chapters in the prescribed textbook and when you prepare for the examination.

You will also receive a separate additional tutorial letter (Tutorial Letter 103) entitled "Mathematical background". This tutorial letter contains notes on certain mathematical notions that you might need in order to comprehend both the second- and the third-year study material to the highest possible degree. At second-year level the contents of Tutorial Letter 103 and the examples of a more mathematical nature in your textbook are not examinable. We do recommend, however, that you have a look at the contents of Tutorial Letter 103 and some of these examples to make you aware of different applications of the contents of this module.

You should therefore, after registration, be in possession of the following pieces of study material:

- Tutorial Letter 101: Welcoming tutorial letter for COS2661, containing essential information,
- Tutorial Letter 102, and
- Tutorial Letter 103: Mathematical background.

We hope you will enjoy your studies in this second-level module and we already look forward to welcoming you to the third-level Formal Logic module.

Best wishes for your studies.

PRESCRIBED SECTIONS

Prescribed sections of the textbook

Introduction
Chapter 1: 1.1 – 1.4
Chapter 2: 2.1 – 2.5
Chapter 3: 3.1 – 3.7
Chapter 4: 4.1 – 4.6
Chapter 5: 5.1 – 5.4
Chapter 6: 6.1 – 6.6
Chapter 7: 7.1 – 7.4
Chapter 8: 8.1, 8.2, 8.4
Chapter 9: 9.1 – 9.6
Chapter 10: 10.1 – 10.4
Chapter 11: 11.1 – 11.5, 11.7 – 11.8
Chapter 12: 12.1 – 12.4
Chapter 13: 13.1 – 13.3, 13.5
Chapter 14: 14.1

MODULE OVERVIEW

The subject of this module is logic. The module represents an introduction to logical consequence as a central concept of logic. Logical consequence can be defined as a relation between a given set of truth-preserving facts and the facts that logically follow from that set. The primary aim of logic is to show us the justifications that allow us to draw conclusions on what follows logically from what, and what doesn't.

We shall study a symbolic language called First-Order Logic (FOL) to develop an understanding of logical consequence. This language, together with the methods of proof and counterexample, will enable us to investigate methods for ascertaining whether or not one FOL sentence follows from others. A proof is a rigorous, step by step justification showing that the conclusion of an argument is (or is not) a logical consequence of the set of the argument's premises. Hence, in order to build a proof one must show whether or not the relation of logical consequence holds between an argument's premises and its conclusion.

GOALS OF THIS MODULE

This course is a study of the language of first-order logic (FOL). Some of the reasons for studying FOL are:

- Our intrinsic interest in the central concept of logical consequence. Logical consequence is a relation among truth-preserving propositions. It often happens that the truth of one or more propositions guarantees the truth of another. When this happens, we say that the proposition whose truth is guaranteed is a consequence of its guarantor(s).
- To develop better, critical reasoning ability. If we manage to show in a rigorous, step-by-step way that the set of an argument's truth-preserving premises justifies the argument's conclusion, we should be able to evaluate our own reasoning accordingly.

The goal of this course is to develop students' ability to identify and critically analyze arguments. Widely understood, logic is the evaluative study of argumentative reasoning. An argument is understood to be a claim that one set of statements, called "premises", provides reason for asserting another, called the "conclusion". As critical endeavors, both formal and informal logic analyze the relationship between premises and conclusion. Informal logic is the critical analysis of natural language arguments. Formal logic, in contrast, analyzes the forms or structures of arguments by developing specialized languages in which the basic structural relationship between premises and conclusion is precisely symbolized and evaluated.

Propositional logic studies the use of various connectives and operators such as 'and', 'or', 'if , then', and 'It is not the case that . . .' in arguments constructed from simple statements. Predicate logic, in contrast, studies the use of various quantifiers such as 'some', 'at least', 'none', and 'every' in arguments that pick out things and what can be said about them. By providing resources for evaluating arguments, logic promises to be a tool for exposing views that cannot be held "for good reasons". For logic to enhance your critical reasoning abilities as students, you should make the effort to examine your views by articulating your reasons for holding them. Logic is, therefore, a normative study of how we should reason.

INTRODUCTION

This introductory chapter gives you a couple of reasons for studying logic and for learning the language of first-order logic FOL. The most important of these are (1) and (2) below.

(1) FOL is directly relevant for a number of sciences, like mathematics and computer science, in particular for artificial intelligence research.

In Tutorial Letter 103 (which contains the mathematical background to the second- and third-level modules in Formal Logic) you will see how the proofs of elementary mathematical results are typically straightforward applications of the proof rules of FOL. Although it will not be possible during this module to discuss the more advanced applications of first-order logic at length, if you are a computer science student, you will surely notice, for example, the similarities between some sentences of FOL and the Boolean expressions of the programming languages that you already know. Furthermore, the concept of logical consequence is of central importance in this module. Whether a given sentence is a valid or is not a valid consequence of a set of given sentences, or more generally, what information (sentences) can correctly be deduced from a given database, is also a central question in many kinds of computer applications, for example in expert systems.

(2) FOL can be used for making informal notions like meaning and truth more precise. This makes it philosophically relevant and useful for linguistics.

By replacing the imprecise notions of everyday language by something more rigorous, FOL teaches us new things about natural language. In particular, it can teach us something about the relationship between a language and the world via the notion of a model. The language of Tarski's World will be the most important special case of a first-order language during this course, and the models with which this language is associated will be the kind of worlds which appear in the world window when the program Tarski's World is run. This will make the notion of a model very easy to understand.

Further, you will see during this course that some sentences of a natural language can easily be translated into FOL and some cannot. In the latter case, FOL can help us understand the structure of natural language better; it can help us see the subtleties and ambiguities, which prevent us from giving precise translations into FOL.

As this introduction states, your prescribed book does not put the main emphasis on translating from English into FOL and back, but rather tries to teach you to use the language FOL first. That is, you will first be taught how to express ideas using the various aspects of the language FOL. The problems of translating into this language will be discussed only when its relevant aspects are understood.

The package to be used in our study of Logic includes several pieces of software:

- Tarski's World 5.0, a new version of the popular program that teaches the basic first-order language and its semantics;
- Fitch, a natural deduction proof environment for giving and checking first-order proofs;
- Boole, a program that facilitates the construction and checking of truth tables and related notions (tautology, tautological consequence, etc.).

SECTION A: PROPOSITIONAL LOGIC

AIMS AND OBJECTIVES OF SECTION A

- Introduce you to the concept of logical consequence, and other notions central to logic such as argument, consistency, and logical truth.
- Introduce you to the basic methods of proof (both formal and informal), and teach you how to use them.
- Introduce you to the formal structure of the language or languages that we use daily in order to deepen your understanding of the meaning of words and of the nature of truth.
- Provide those of you who will encounter formal logic outside of this course with a firm foundation in symbolic logic. (This includes students of philosophy, as well as students of mathematics and computer science disciplines.)

Chapter 1: Atomic Sentences

Outcomes of the chapter

- Understand the concept of formal first-order languages.
- Know the syntax of FOL: predicate symbols, individual constants, function symbols.
- Are acquainted with examples of first-order languages: the blocks language, the language of arithmetic.

This chapter begins the discussion of FOL by introducing this language's *atomic sentences*. Although FOL can contain a much larger vocabulary than what we will use, we will mostly concentrate on the portion that can be used to describe the worlds we can build using the software program Tarski's World.

At the very simplest level, FOL contains sentences that contain two kinds of things:

- individual constants (names) and
- predicate symbols (property and relation words).

More will be said about atomic sentences in the next sections.

1.1 Individual constants

Individual constants are simply symbols that are used to refer to some fixed individual object. They are the FOL analogue of names. As is stated in this section, the individual constants of FOL are allowed to consist of several letters (like max or claire) and of numbers (like 1) so that it is easier to remember what they mean. They are written in lower case letters.

A difference between names in English and the individual constants of FOL is that the individual constants refer to exactly one object (while in English, for example, more than one person may have the same name). However, an object can have more than one name or no name at all.

1.2 Predicate symbols

Just like the individual constants, the predicate symbols of FOL are allowed to have more letters than just one so that it is easier to remember what each of them means. Predicate symbols are symbols used to denote some property of objects or some relation between objects. Consider the sentence

Claire gave Scruffy

In FOL, we view this as a claim that the predicate *Gave* expresses a relation between the individuals referred to by the names *claire* and *scruffy*. These individuals are called the "arguments" and in this case, there are two.

Every predicate symbol comes with a single fixed ‘arity’, a number that tells you how many names it needs to form an atomic sentence.

The term ‘arity’ comes from the fact that predicates:

- Taking one argument are called unary e.g. Cube, Tet, Medium, etc
- Those taking two are called binary e.g. Smaller, LeftOf, SameSize, SameCol, Adjoins, = etc.
- Those taking three are ternary e.g. Between.

A predicate in FOL is always a single word, in the sense that it does not contain any spaces.

1.3 Atomic sentences

Atomic sentences correspond to the simplest declarative sentences of English, and often consist of names and predicates. An example of such an English sentence is:

Claire is taller than Max
with its first-order logic (FOL) translation
Taller(claire, max)

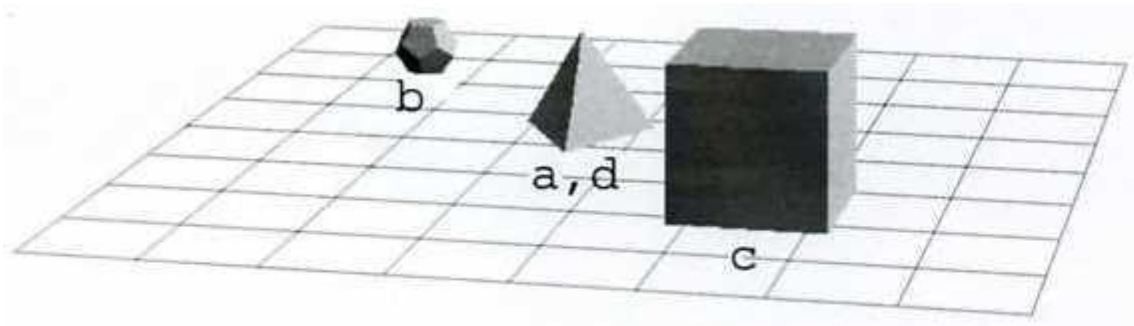
We will begin all predicates in FOL with a capital letter, but different versions of FOL have different conventions for forming names and predicates and different conventions for combining them.

A new notation called the *infix notation* is introduced in this section. In this notation a two-place predicate symbol is placed between its arguments, like the symbol = has been placed in "a=b”. All the other predicates used are in *prefix notation,* like Cube(a), SameSize(a, b) and Between(a, b, c) etc.

The order of names in an atomic sentence is quite important eg Taller(claire, max) is different from Taller(max, claire). Predicates and names designate properties and objects respectively. What makes sentences special is that they make claims (or express propositions). A claim is something that is either true or false, which of these it is we call its truth value. Thus Taller(claire, max) expresses a claim whose truth value is (say) True, while Taller(max, claire) expresses a claim whose truth value is False.

Exercise 1.3

You are asked to build a world in which ten given sentences in the language of Tarski's World are all simultaneously true. Here is such a world, firstly as it appears in Tarski's World and secondly in the form of a table.



back

		b: D, S							
left				**a**,**d**: T, M					right
						c: C, L			

front

KEY:

b: D, S	Dodecahedron, small, named b
a, d: T, M	Tetrahedron, medium, two names namely a and d
c: C, L	Cube, large, named c

A few hints about drawing up worlds to make a set of sentences true:

1. Look down the whole list of sentences counting names and noticing whether any of the sentences is an identity statement. This will tell you *how many objects* you are dealing with. Remember that every name in a sentence must refer to an object in the world but not every object in the world need have a name. At this stage of the course, before you come to quantifiers, the only way to refer to an object is by a specific name, so the number of names pretty well gives you the number of objects (allowing for identity statements where one object has two names).

2. Next, start to furnish the world by processing the unary predicates (i.e., those that take only one argument or name): Tet, Cube, Dodec, Medium, etc. In Exercise 1.2 block **a** is specified as a medium tetrahedron, **b** as a dodecahedron and **c** as a cube.

3. You can fill in the information given in the binary predicates now: =, Smaller, Larger, FrontOf and LeftOf, as in sentences 5, 7, 8, 9 and 10.

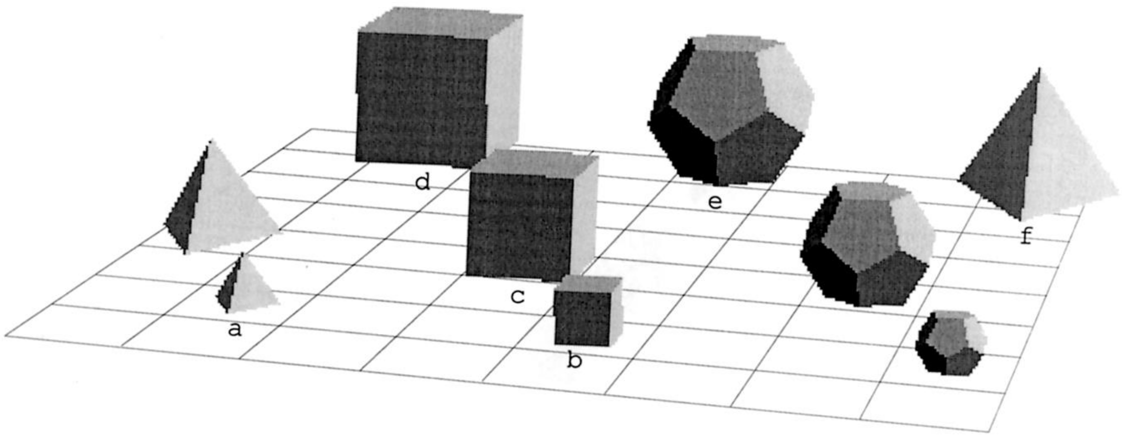
4. Finally, if you have the ternary predicate Between (sentence 6 here) in the list of sentences, make sure your world complies with the claim it makes.

5. You may find that some name, say '**e**', may appear only in a sentence like Large(e) . It's up to you then whether to make it a cube, a dodecahedron or a tetrahedron. Similarly, if there is no sentence giving the size of some object, either directly in a unary predicate or indirectly by saying it is Larger or Smaller than some other object whose size is known, then you can decide its size for yourself. The list of sentences given in exercise 1.2 consists of atomic sentences only. If you have to construct a world from a list containing complex claims as well, i.e., sentences with connectives in them, you should start by processing the atomic claims, then claims with two atomic sentences in them, then three, etc., with due regard to the rule for the particular connectives that link them.

Exercise 1.7

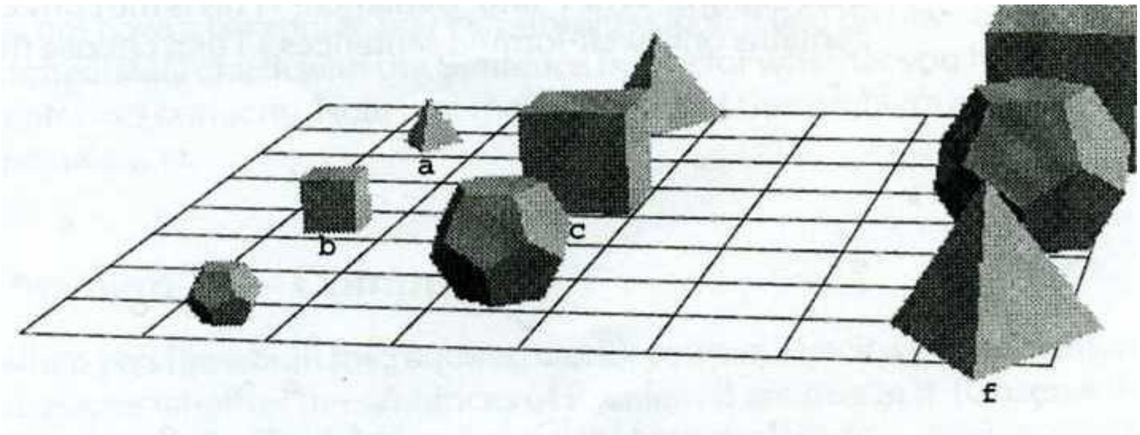
Read carefully through this exercise, as it is formulated in the book on page 27. The file AUSTIN.SEN which is mentioned in it contains the following six sentences:

1. Larger(a, b)	2. Between(c, d, b)	3. LeftOf(c, b)
4. FrontOf(c, b)	5. BackOf(e, b)	6. RightOf(a, b)

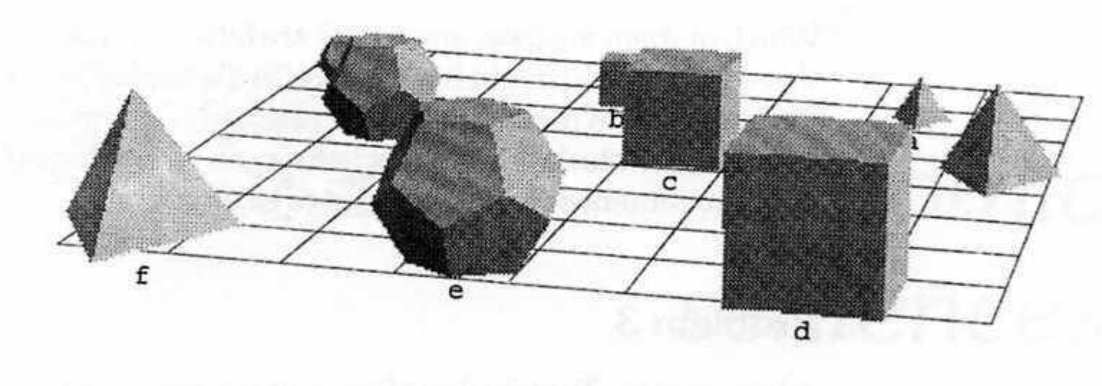


Wittgenstein's world

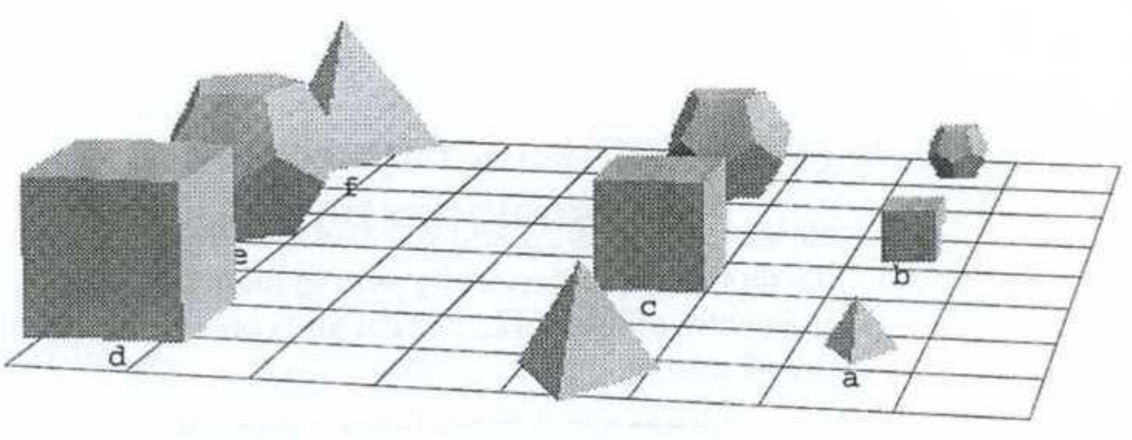
Wittgenstein's world, which is located in the file WITTGENS.WLD, is shown above. Next you will be shown its three rotated versions, each one obtained from the earlier one by rotating the display by 90°. Evaluate each of the above sentences in each of these four versions and write down the results in a table. Use your table to answer the questions in the last paragraph of the problem as it is given in your textbook.



Wittgenstein's world rotated by 90°



Wittgenstein's world rotated by 180°



Wittgenstein's world rotated by 270°

Solution to exercise 1.7:

The sentences in **Austin.sen** have, in the four cases that are under consideration, the following truth values:

Sentence	Rotation of 0°	Rotation of 90°	Rotation of 180°	Rotation of 270°
1. Larger(a,b)	FALSE	FALSE	FALSE	FALSE
2. Between(c,d,b)	FALSE	FALSE	FALSE	FALSE
3. LeftOf(c,b)	TRUE	FALSE	FALSE	TRUE
4. FrontOf(c,b)	FALSE	FALSE	TRUE	TRUE
5. BackOf(e,b)	TRUE	FALSE	FALSE	FALSE
6. RightOf(a,b)	FALSE	FALSE	TRUE	FALSE

1.4 General first-order languages

The symbols of FOL are thought of as having fixed meanings. It is for this reason that one constructs a new version of FOL for each situation of a new kind. In other words, various versions of FOL are *interpreted*. As you will learn in the next chapter, this feature of FOL also affects the definitions of the *satisfiability* and the *logical truth* of its sentences. Each of these notions can be understood in two ways in an interpreted language.

Answers to exercise 1.9 page 30

With reference to Table 1.2 on page 30, the translations of the English sentences are:

1. Owned(claire, folly, 2:00)
2. Gave(claire, pris, max, 2:05)
3. Student(max)
4. Fed(Claire, carl, 2:00)
5. Owned(max, folly, 3:05)
6. 2:00 < 2:05

Answers to exercise 1.10 page 31

1. Max owned Scruffy at 2 pm, Jan 2, 2001.
2. Max fed Scruffy at 2:30 pm, Jan 2, 2001.
3. Max gave Scruffy to Claire at 3 pm, Jan 2, 2001.
4. 2 pm, Jan 2, 2001 is earlier than 2 pm, Jan 2, 2001.

Note: Read sections 1.5 to 1.8 of the prescribed book, but do not study. These sections are not examinable.

Chapter 2: The Logic of Atomic Sentences

Outcomes of the chapter

Understand logical validity of arguments.

- Know how to show that arguments are valid.
- Understand the basic properties of the identity predicate: reflexivity, principle of the substitutability of identicals.
- Understand the basic properties of other predicate symbols (transitivity, reflexivity, symmetry, inverse relations).
- Able to construct informal proofs.
- Able to use Fitch and construct formal proofs.
- Know how to show that arguments are not valid: the method of counterexamples.

2.1 Valid and sound arguments

This section introduces the notions of *logical validity, logical consequence* and *soundness* of deductive argument. It is important to know when all these notions are present in an argument. What is an argument? It is a group of propositions or a set of statements of which one (the conclusion) is meant to follow from the others (premises).

Validity

An argument is logically valid if and only if it takes a form that makes it impossible for the premises to be true but the conclusion nevertheless to be false. Validity requires that there be no possible circumstance in which the premises would be true and the conclusion false. One can also say that the conclusion is a logical consequence of the premises.

An example of a valid argument can be the following:

All women are mortal
Margaret is a woman
∴ Margaret is mortal

In Fitch format the argument is given as:

| All women are mortal
| Margaret is a woman
|—
|
| Margaret is mortal

There are cases where we have valid arguments with a false conclusion. This occurs when there is at least one premise which is false (otherwise the argument would not be valid). If an argument has only true premises but a false conclusion, it is regarded as an invalid argument.

Soundness

A deductive argument is *sound* if and only if

- it is both valid, and
- all of its premises are *actually true*.

The above argument is a good example of a sound argument.

Partial answer to exercises 2.1 – 2.3 page 44

Argument	**Valid?**	**Sound in Socrates' world**	**Sound in Wittgenstein's world**
1	Yes	Yes	No
2			
3			
4	Yes	Yes	No
5			
6			
7	No	No	No
8			

You may complete the table yourself.

Answers to Exercise 2.4 page 46

a) The question asked whether a valid argument can have false premises and a false conclusion.

Yes: Suppose a teacher gives chocolates to all the girls in the class and chips to all the boys. Suppose Andy is a boy. Then the following is a valid argument with one false premise (the third) and a false conclusion:

The teacher gives chocolates to all the girls in the class.
The teacher gives chips to all the boys in the class.
Andy is a girl.
Andy gets chocolates.

b) The question asked whether a valid argument can have false premises and a true conclusion.
Yes: Suppose a teacher gives chocolates to all the girls in the class and to one boy. Suppose Andy is the boy who gets chocolates. Then the following is a valid argument with one false premises (the second) and a true conclusion:

| The teacher gives chocolates to all the girls in the class and to one boy.
| Andy is a girl.
|—
| Andy gets chocolates.

c) Can a valid argument have true premises and a false conclusion?

No! An argument is valid if and only if the truth of its premises forces the truth of its conclusion.

d) Can a valid argument have true premises and true conclusion?

Yes: Suppose a teacher gives chocolates to all the girls in the class and chips to all the boys. Suppose Michael is a boy. Then the following is a valid argument with true premises and a true conclusion:

| The teacher gives chocolates to all the girls in the class.
| The teacher gives chips to all the boys in the class.
| Michael is a boy.
|—
| Michael gets chips.

2.2 Methods of proof

Some other important concepts which are introduced in this section are those of a *formal proof* and an *informal proof*. An informal proof expresses in English the same argument that a formal proof would express in a language of logic.

As you are told in this section, in an informal proof one often leaves some of the more obvious steps out. You might now wish to ask which steps it is that you are allowed to leave out - after all, many of the things that we prove during this course are pretty obvious and you might wonder whether you could omit all the intermediate steps of a proof, claiming that they are all obvious. The answer is that this won't do: in a logic course also the informal proofs are expected to be rather detailed. In particular, they must be much more detailed than the proofs that mathematicians normally write. Your informal proofs do not have to be quite as detailed as the informal proofs that you will find in the solutions to the exercises and problems, but it is nevertheless a good idea to use these proofs as models when you write down your own informal proofs.

(For exam purposes: always remember to double-check whether you were asked to give a *formal* or an *informal* proof. If you are asked for an informal proof, but you are unsure about the degree of detail expected from you, or you have no idea how to begin [unlikely in any case], remember giving a formal proof instead may still earn at least some marks. *But* if you are asked to give a formal proof and give an informal one, you will receive *no* marks at all.)

This section also provides you with some examples of proof rules that can be applied to atomic sentences. These include the reflexivity of identity and the transitivity of the relation "less than". You should observe that these proof rules make sense only in an *interpreted language*, in a language in which the predicate and constant symbols are assumed to have some particular meanings. For example, the transitivity of the relation "less than" implies that it is legitimate to infer from $k_1<k_2$ and $k_2<k_3$ that $k_1<k_3$.

This rule makes sense only when "b<c" means that b is less than c, but not necessarily otherwise; if we decided that "b<c" meant (let's say) that b is two times c, the rule would be false. In this respect the transitivity of "less than" is different from the inference rules that you have learned during the first-year course. These rules also remain valid when the meanings of predicates are changed. For example, according to the rule of disjunction introduction it is legitimate to infer from ($k_1<k_2$) that ($k_1<k_2$) $\vee$ ($k_2<k_3$). The validity of this step would not be affected if we changed the meaning of the predicate symbol "<". As a matter of fact, most logic textbooks work with languages whose predicates are assumed to be uninterpreted with the exception of the identity predicate. In such books the transitivity of "less than" would not be called an inference rule of logic. We shall return to the significance of this special feature of the textbook that we are using in the additional notes to the next chapter.

There are four important principles that hold of the identity relation:

*Read the **Remember** section on page 51 of the prescribed book.*

Answer to exercise 2.6 page 53

We are told that SameRow(a, a), that a = b, and that b = c. From the latter two we can conclude a = c by transitivity of identity. Now substitute the name 'c' for the first use of the name 'a' in SameRow(a, a), using the indiscernibility of identicals. This gives the desired conclusion SameRow(c, a).

Answers to exercise 2.7 page 53

a) *Does (3) follow from (1) and (2)?*

Yes: (3) does follow from (1) and (2), because Nancy is Max's mother and since Max and Claire are not related, Nancy cannot be Claire's mother.

b) *Does (2) follow from (1) and (3)?*

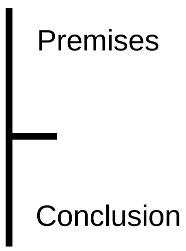
No: (2) does not follow from (1) and (3). We can imagine a situation where Max and Claire are nor related and Nancy is neither's mother. In such a world (1) and (3) are true, but (2) is false, so it is not a consequent of those two.

c) *Does (1) follow from (2) and (3)?*

No: (1) does not follow from (2) and (3). We can imagine a situation where Nancy is Max's mother, but not Claire's, and Claire and Max are cousins. So (2) and (3) become true, but (1) false.

2.3 Formal proofs

This section introduces the system ***ƒ***, the deductive system within which formal proofs are constructed in your textbook. There are two rules in the system ***ƒ*** corresponding to each connective and to each quantifier, an introduction rule and an elimination rule. In the system ***ƒ*** a graphical device is used which has horizontal and vertical bars. The vertical line that runs on the left of the steps shows us that we have a single purported proof consisting of a sequence of several steps, and the horizontal Fitch bar indicates the division between the claims that are assumed and those that allegedly follow from them:



In a formal proof in ***ƒ***, we use the Fitch bar notation above. The premises are written above the (horizontal) Fitch bar; the subsequent steps (intermediate conclusions and the ultimate conclusion) are written below the Fitch bar.

Each step in a formal proof must be entered in accordance with some precisely stated rule of the formal system of rules. By applying a rule to some previous line or lines in a proof, we provide a justification for entering a new step in a proof.

IDENTITY RULES

a) Identity Introduction (= Intro)

This rule says, in effect, that you may enter a sentence of the form n = n at any point you wish. Obviously, this rule embodies the principle of reflexivity of identity.

Diagram given on page 55 of the prescribed book

b) Identity Elimination (= Elim)

This rule of **= Elim** tells you that you may substitute m for n wherever you like, provided that you have the sentence n = m. This rule embodies the principle of indiscernibility of identicals.

Notice that although the rule is called an “elimination” rule, nothing is really being eliminated. The idea is that we have used (eliminated?) an identity sentence in the process of arriving at a conclusion. That is, we are arguing “from” an identity sentence, and in that sense we are “eliminating” it.

Diagram given on page 56 of the prescribed book

Therefore, the **= Intro** tells us how to enter an identity sentence, e.g. we can enter n = n, and **= Elim** tells us how to use an identity sentence (n = m) as a premise. The other identity rules, the symmetry and transitivity of identity follow from reflexivity and indiscernibility. That is, using only **= Intro** and **= Elim**, you can prove that b = a follows from a = b, and that a = c follows from a = b and b = c.

Reiteration: (Reit)

This rule tells us that at any point in a proof you may enter any sentence that occurs on some previous accessible line. P occurs above the entry point (the small triangle) and is entered at the entry point in accordance with the rule of Reiteration. You may *not*, however, use Reiteration to "reiterate" a sentence that occurs (only) within the scope of an inaccessible subproof (which will be discussed later).

2.4 Constructing proofs in Fitch

Work out the "You try it' on page 58-59.

In the exercises you are required to use Fitch to construct a formal proof that the conclusion is a consequence of the premises.

Solution to exercise 2.18 page 62

1. Between(a, d, b)	
2. a = c	
3. e = b	
4. Between(c, d, b)	= Elim: 1, 2
5. e = e	= Intro
6. b = e	= Elim: 5, 3
7. Between(c, d, e)	= Elim: 4, 6

2.5 Demonstrating nonconsequence

You have read that there is a very different, but equally important kind of proof, which is a proof of nonconsequence. This means we show that a conclusion does *not* follow from the premises, i.e. that it would be an invalid argument. (Remember that an argument is invalid when it is possible for premises to be true and the conclusion false.) In this a counterexample has to be found.

An informal proof of nonconsequence describes a possible situation in which the premises are true and the conclusion false, in as much detail as is necessary to make it clear that it is really a possible situation.

Formal proofs of nonconsequence in the blocks language use Tarski's World to construct a world where the premises are True and the conclusion is False.

Answer to exercise 2.22 page 65

The argument is invalid. Hence it cannot be sound, because sound arguments are a special kind of valid argument, one where the premises are all true. To see that it is not valid, imagine that there were only two computer scientists, Bill Gates and Steve Jobs, that they were both rich, but that only Steve Jobs knows how to program. Then all the premises would be true, but the conclusion false. So the argument cannot be valid.

Now try doing exercise 2.23 on your own.

Chapter 3: The Boolean Connectives

Outcomes of the chapter

- Understand the syntax, semantics and truth tables of Boolean connectives.
- Know the formation rules for sentences of FOL using ∧, ∨, ¬.
- Be able to translate sentences from English into FOL using these Boolean connectives.
- Understand the expressive power of these Boolean connectives.
- Be able to express complicated statements using the blocks language and these Boolean connectives.

This is a chapter on the three connectives ¬, ∨ and ∧. These are truth-functional connectives, in which the truth value (truth or falsity) of a compound sentence formed with such a connective is a function of (i.e., is completely determined by) the truth values of its components. An interesting way to deal with these connectives is given in the form of a game, sometimes called the Henkin-Hintikka game.

3.1 Negation symbol: ¬

In the game facility of Tarski's World there are two players, namely the computer and its user. At each stage of the game there is a sentence which is under consideration, and the players are committed to opposite truth-values for this sentence. That is, one of them claims that the sentence is true and the other one that it is false. The move that is made in the game depends on the *main connective* of the current sentence. Normally at each stage of the game one of the players has the possibility to choose between two or more alternatives.

However, the case where the main connective of the sentence is a negation is an exception. In this case there is nothing to choose. What happens in this case is that the negation symbol is simply removed from the sentence and the commitments of the players are swapped.

Answers to exercise 3.2 page 70

1. True
4. False
5. This is false since it claims that **a** and **b** are not in the same row, but they are in the same row in the given world.
8. ¬¬LeftOf[a, a] is logically equivalent to LeftOf[a, a], which is false in any world that has a block named **a**, so it is false in Boole’s World.

3.2 Conjunction symbol ($\wedge$)

As is explained in this section the game proceeds as follows when the main connective of the considered sentence is a conjunction. The player who is committed to the falsity of the conjunction chooses one of its conjuncts. After that she is taken to be committed to the falsity of the conjunct she has chosen, and the other player will be taken to be committed to its truth.

3.3 Disjunction symbol ($\vee$)

The way in which the game proceeds when the main connective of the sentence is a disjunction is analogous to the case in which it is a conjunction. In this case the player who has committed herself to the truth of the sentence must choose one of the disjuncts. She will then be taken to be committed to its truth and the other player will be taken to be committed to its falsity.

3.4 Remarks about the game

The game rules are given so you do not need to memorize them, because Tarski's World will always tell you what your commitments are (after you choose your initial commitment), and will tell you when it is your turn to move.

To play the game and be sure of winning, you will need to know not only that a sentence has the truth value you say it has (your commitment), but also why it does. This means, for example, that if you know that a disjunction is true, you will need to know which disjunct is true in order to be sure of winning. Similarly, if you know that a conjunction is false, you will need to know which conjunct is false in order to be sure of winning.

3.5 Ambiguity and parentheses

This section discusses the use of parentheses in the language of Tarski's World and in the other versions of FOL that are used in the book. As it is stated here, in these languages you are in some cases allowed to leave out parentheses if this does not create ambiguity. By the way, one could reduce the number of parentheses even more without causing ambiguity if one introduced a fixed "order of preference" for the connectives. Such an order would provide us with rules that were analogous with the rules of algebra which tell us, for example, that the expression a+b×c means a+(b×c) rather than (a+b)×c. Similarly, we could decide (let's say) that $\wedge$ comes before $\vee$ so that:

> Home(max) $\vee$ Home(claire) $\wedge$ Happy(carl) means
>
> > Home(max) $\vee$ (Home(claire) $\wedge$ Happy(carl))
>
> rather than
>
> > (Home(max) $\vee$ Home(claire)) $\wedge$ Happy(carl).

Some logic textbooks introduce an ordering like this, but your textbook does not. That is why "Home(max) $\vee$ Home(claire) $\wedge$ Happy(carl)" does not qualify as a well-formed sentence during this course.

3.6 Equivalent way of saying things

This section introduces the symbol of logical equivalence ⇔. At this point it is absolutely essential to understand that the symbol ⇔ is *not a part of FOL*. When we write P ⇔ Q, we are *talking about* a language of first-order logic. The sentence P ⇔ Q is short for the *English* sentence "P and Q are logically equivalent", but it is *not itself* a sentence of FOL. The language in which one talks about a language of logic is often called the *metalanguage* and thus, for example, it is said that the symbol ⇔ is a part of the metalanguage in which we talk about FOL.

3.7 Translation

This section names a condition that a translation between English and FOL must satisfy in order to be an acceptable one: an English sentence and a FOL sentence must have *the same truth value in all circumstances*, if one of them is to count as *an acceptable translation* of the other one. This condition is clearly *necessary*, and usually it can also be accepted as a *sufficient* one, as one does in this section. However, the idea that, whenever a FOL sentence always has the same truth value as an English sentence, each of them can be translated by the other one, has some very odd consequences if one sticks to it. For example, consider a case in which the English sentence happens to be a tautology or a contradiction. If the original English sentence is - let's say - "b is large or b is not large" then *any tautology whatsoever* of FOL has in all circumstances the same truth value as this sentence ("b is large or b is not large") has. For example, Tet(a) ∨ ¬Tet(a) always has the same truth value as the above sentence ("b is large or b is not large") has, precisely because both of these sentences are always true. However, it sounds very strange to say that the latter sentence is a good translation of the former one, and - by the way - such a "translation" would not be considered acceptable in the examination. Nevertheless, a good way to test whether a given translation is acceptable is to try to figure out whether the two sentences have always the same truth value.

Answer to exercise 3.25 page 89

First, we must specify the version of FOL that we are using. This is what has been done in the table below. Using the version of FOL which is specified in the table, the sentences can be translated as follows:

1. LessContagious(aids, influenza) ∧ MoreDeadly(aids, influenza)
2. Fooled(abe, stephen, sunday) ∧ ¬Fooled(abe, stephen, monday)
3. (Admires(sean, meryl) ∧ Admires(sean, harrison)) ∨ (Admires(brad, meryl) ∧ Admires(brad, harrison))
4. Jolly(daisy) ∧ Miller(daisy) ∧ River(dee) ∧ Lives(daisy, dee)
5. ¬(Borrower(eldestChild(polonius)) ∨ Lender(eldestChild(polonius)))

ENGLISH	FOL	COMMENTS
Names:		
Abe	abe	the name of a person
AIDS	aids	the name of a disease
Brad	brad	the name of a person
Daisy	daisy	the name of a person
Harrison	harrison	the name of a person
Meryl	meryl	the name of a person
Dee	dee	the name of a river
Sean	sean	the name of a person
Influenza	influenza	the name of a disease
Monday	monday	the name of a day of the week
Polonius	polonius	the name of a person
Stephen	stephen	the name of a person
Sunday	sunday	the name of a day of the week
Predicates:		
x admires y	Admires(x,y)	
x is a borrower	Borrower(x)	
x fooled y on z	Fooled(x,y,z)	Here z is a day.
x is jolly	Jolly(x)	
x is a lender	Lender(x)	
x is less contagious than y	LessContagious(x,y)	
x lives near y	Lives(x,y)	
x is a miller	Miller(x)	
x is more deadly than y	MoreDeadly(x,y)	
x is a river	River(x)	
Functions:		
eldest child of x	eldestChild(x)	

Chapter 4: The Logic of Boolean Connectives

Outcomes of the chapter

- Understand logical truth, tautologies, and TW-necessities.
- Understand tautological equivalence, consequence, and validity.
- Able to apply the method of truth tables.
- Understand and able to show tautological equivalences: De Morgan's Laws and other equivalent transformations.
- Able to prove tautological equivalence by a chain of equivalences.
- Able to transform formulas to negation, conjunctive and disjunctive normal forms.

The chapter discuss what truth tables can tell us about three related logical notions which are:

- logical consequence
- logical equivalence and
- logical truth

In these sections relations between a number of general concepts and ways to construct precise (but incomplete) analogs using truth tables are given. The truth-table analogs are more restrictive than their general counterparts. But they have the virtue of being precise, clearly specified, and in fact algorithmic. On the other hand, although constructing and checking a truth table provides an algorithm for determining tautology, tautological equivalence, and tautological consequence, the algorithm is not very efficient.

4.1 Tautologies and logical truth

Logical Truth or Necessity: A sentence that is a logical consequence of any set of premises. That is, no matter what the premises may be, it is impossible for the conclusion to be false (e.g. a = a).

Logical Possibility: A sentence or claim is logically possible if there is a possible circumstance in which it is true.

Answer to exercise 4.9 page 105

Sentence	**TT-Possible**	**TW-possible**
1	Yes	Yes
2	No	No
3	Yes	No
4	Yes	No
5	Yes	Yes
6	Yes	No
7	Yes	No
8	No	No
9	Yes	Yes
10	Yes	Yes

4.2 Logical and tautological equivalence

Logical equivalence
Two sentences are logically equivalent if they have the same truth value in all *possible* circumstances.

Tautological equivalence
S and S' are tautologically equivalent if and only if every row of the joint truth table assigns the same values to S and S'.

4.3 Logical and tautological consequence

Logical Consequence:
A sentence S is a logical consequence of a set of premises if it is impossible for the premises all to be true while the conclusion is false.

Tautological Consequence:
Q is a tautological consequence of a set of premises P_1, P_2, ..., P_n if and only if every row in a truth table that assigns T to P_1, P_2, ..., P_n also assigns T to Q.

Truth tables:

a) The number of rows in the table for a given sentence is a function of the number of atomic sentences it contains.
b) If there are *n* atomic sentences, there are 2^n rows. Each row represents a possible assignment of truth values to the component atomic sentences.
c) On each row, the values of the atomic sentences determine the values of the compounds of which they are components.

d) The values of the compounds of atomic sentences in turn determine the values of the larger compounds of which they are components.

e) Finally a unique value for the entire sentence is determined on each row.

A table below summarizes the consequences and the equivalences given in the above sections.

General logical concept	Truth-table analog
Logical truth/necessity P is logically necessary iff it is impossible for P to be false	***Tautology*** P is a tautology iff it is true in every row of its truth table
Logical equivalence P and Q are logically equivalent iff it is impossible for it to be the case that one of them is true and the other false	***Tautology equivalence*** P and Q are tautologically equivalent iff there is no row of their joint truth table in which one is true and the other false
Logical consequence C is a logical consequence of P_1, P_2, ..., P_n iff it is impossible for P_1, P_2, ..., P_n to all be true and C to be false	***Tautological consequence*** C is a tautological consequence of P_1, P_2, ..., P_n iff there is no row of their joint truth table in which P_1, P_1, P_2, ..., P_n are all true and C is false

4.4 Tautological consequence in Fitch

Truth tables are mechanical i.e. there is an automatic procedure (an “algorithm”) that always give you an answer to the question whether a sentence is a tautology or whether a sentence is a tautological consequence of another sentence or set of sentences. Another method of checking whether a sentence is a tautological consequence is by using the program Fitch.

*Do the **You try it** on p. 114 to see how to do this.*

Taut Con, FO Con, and Ana Con

These are three methods, of increasing strength that Fitch uses to check for consequence.

- **Taut Con** checks to see whether a sentence is a tautological consequence of some others. It pays attention only to the truth functional connectives. It is the weakest procedure of the three because it only catches tautological consequence, and misses the broader notions of consequence.
- **FO Con** checks to see whether a sentence is a “first-order” consequence of some others. It pays attention not only to the truth functional connectives but also to the identity predicate and to the quantifiers.
- **Ana Con** checks to see whether a sentence is an “analytic” consequence of some others. It pays attention not only to the truth functional connectives, the identity predicate, and the quantifiers, but also to the meanings of most (but not all!) of the predicates in the blocks language. This notion comes the closest of the three to that of (unrestricted) logical truth.

If a sentence is a tautological consequence of some others it is clearly also a first-order consequence and an analytic consequence of those sentences. But the converse does not hold - some first-order consequences are not tautological consequences, and some analytic consequences are not first-order consequences.

A warning about Ana Con

The **Ana Con** mechanism does not distinguish between logical necessity and TW-necessity. That is, it counts at least some "Tarski World" consequences as analytic consequences along with logical consequences more narrowly conceived.

An example:

According to **Ana Con**, Cube(b) is an analytic consequence of ¬Tet(b) ∧ ¬Dodec(b). This is not a first-order consequence, and hence not a tautological consequence either. This happens because **Ana Con** pays attention not only to the meanings of some of the predicates, but also to some of the special features of Tarski's World. Since in Tarski's World there are only three shapes of blocks, it follows that there cannot be a Tarski World in which an object is neither a tetrahedron nor a cube nor a dodecahedron.

But while that may be true for every Tarski World, it does not hold for every possible world. In general, it does not follow logically, from the fact that b is neither a tetrahedron nor a dodecahedron, that b is a cube - b might be a sphere. So this example does not seem to be a logical necessity, but only something weaker - a TW-necessity.

Ana Con also has some other limitations. It misses certain TW-necessities, namely, those involving the predicates Adjoins and Between, which it does not understand. For example, ¬Large(a) is a TW-consequence of Adjoins(a, b), since it is impossible in a Tarski world for a large block to adjoin another block. But **Ana Con** will not recognize this consequence. Similarly, **Ana Con** does not understand any predicates that are not in the blocks language. Hence, it will not know that Older(b, a) is a logical consequence of Younger(a, b), since these predicates are not in the blocks language. So you must use **Ana Con** with caution!

4.5 Pushing negation around

This section also mentions a number of logical equivalences between sentences of FOL. The *associativity rules* and the *idempotence* and *commutativity rules* on pages 118 to 119 are so obvious that you do not normally have to mention them explicitly when you use them in an informal proof. In addition to these rules, this section contains three other important equivalences:

- *The law of double negation*, and two of
- *De Morgan's laws*. (In addition to these De Morgan's laws, there are also De Morgan's laws for quantifiers. These will be presented on page 279 in section 10.3 of your textbook.)

With these and the other rules that are presented in this section you can construct derivations which show the equivalence of some sentences. Of course, these derivations are not *formal* proofs in the sense of the system ***F*** which is used for constructing formal proofs in FOL.

4.6 Conjunction and disjunctive normal forms

This section is concerned with the distributive laws and with putting sentences into a normal form with the help of these laws. Three kinds of normal forms are discussed: disjunctive normal forms, conjunctive normal forms, and negation normal forms. Observe that, by definition, every sentence which is in conjunctive normal form or in disjunctive normal form is also in negation normal form, but not necessarily *vice versa*.

Chapter 5: Methods of Proof for Boolean Logic

Outcomes of the chapter

- Able to give informal proofs showing that arguments are valid.
- Understand the construction of formal rules for $\wedge$ and $\vee$.
- Able to use proof by cases and proof by contradiction in informal proofs.
- Understand the concept of inconsistent premises.

Truth tables are useful and powerful tools for checking satisfiability, validity and equivalence in simple arguments, but have two major limitations which are:

- Truth tables are difficult and strenuous to use when a large number of atomic sentences are involved.
- Truth tables cannot be easily extended to reasoning where validity depends on more than just truth-functional connectives.

5.1 Valid inference steps

In this section you are made familiar with three methods that you may use in *informal* proofs:

- **Conjunction elimination**
  From P $\wedge$ Q, infer P (or infer Q).
  This pattern of inference is sometimes called *simplification*. From a conjunction, infer any of the conjuncts.
- **Conjunction introduction**
  From P and Q, infer P $\wedge$ Q.
  This pattern of inference is sometimes called just *conjunction*. From a pair of sentences, infer their conjunction.
- **Disjunction introduction**
  From P, infer P $\vee$ Q (or infer Q $\vee$ P)

5.2 Proof by cases

A *proof by cases* corresponds to the rule *disjunction elimination* which will be presented in chapter 6 of your textbook. This rule and the proofs by cases that are presented in this section share the same basic idea, which is that if you know that a disjunction is true and if you know that some sentence P follows from each of the disjuncts, then you may conclude P.

Answer to exercise 5.7 on page 135

Here is an informal proof of the argument.

By the third premise, either Claire is not at home or Carl is happy. Let us break the statement into cases.

- If Carl is happy, then we have the desired conclusion.
- The other case is where Claire is not at home. By the first premise, either Claire or Max is home, so, since we are assuming that Claire is not, then Max is at home. But by the second premise, either Max is not at home or Carl is happy. So it must be that Carl is happy.

So in either case, Carl is happy.

Try to do the other exercises.

5.3 Indirect proof: proof by contradiction

Similarly, the rule of *negation introduction* which will be presented in chapter 6 is a formalisation of the method of *proof by contradiction*. The basic idea behind this method of proof is that, if we manage to prove a contradiction from some premise, then the premise itself must be false. So the method consists of showing that some assumption (together with the argument's premises) leads to a contradiction. The rule is also called *indirect proof* or *reductio ad absurdum* and is a powerful method of proof commonly used in mathematics.

Suppose you wish to establish that a conclusion of the form ¬S is a logical consequence of a set of premises P_1, P_2, … P_n. You assume S (equivalent to the negation of the argument's conclusion) and treat it as a premise along with P_1, P_2, … P_n. You then try to deduce from these assumptions a contradiction, i.e. a pair of sentences that contradict one another, e.g. Q and ¬Q. You may then (no longer assuming S) conclude that ¬S. An example of an indirect proof is given in your textbook page 136.

Negation Introduction: This is the formal counterpart of "proof by contradiction": If you can prove a contradiction ⊥ on the basis of an additional assumption P, then you may infer ¬P from the original premises.

⊥ **Introduction**: This rule allows us to obtain a contradiction symbol if we have established an explicit contradiction in the form of P and its negation ¬P.

⊥ **Elimination**: This rule allows us to assert any sentence P, once we have established a contradiction.

5.4 Arguments with inconsistent premises

If a set of premises is inconsistent, any argument having those premises is valid. The reason is that, if the premises are inconsistent, there is no possible circumstance in which they are all true. So no matter what the conclusion is, there is no possible circumstance in which the premises are all true and the conclusion false.

But no such argument is sound, since a sound argument is not only valid but has true premises. Why be interested in arguments with inconsistent premises? Well, we know that if you can derive a contradiction $\bot$ from a set of premises, the set is inconsistent. (If it were possible for the premises all to be true, then since we have derived $\bot$ from them, it would have to be possible for $\bot$ to be true, and this clearly is not possible.)

We may not know, at the start, that our premises are inconsistent, but if we derive $\bot$ from them, we have established that they are inconsistent. If a set of premises, or assumptions, is inconsistent, it is important to know this. And being able to deduce a contradiction from them is an excellent way of showing this. We may not be able to show, using logic alone, which premise is false, but we can establish that at least one of them is false.

Chapter 6: Formal Proofs and Boolean Logic

Outcomes of the chapter

- Have mastered the basic properties of ¬.
- Understand indirect proof and formal proofs with ¬.
- Understand and able to construct formal proofs involving introduction and elimination of ∧, ∨, ¬ and ⊥.
- Understand and able to use subproofs correctly.

The deductive system ***f***, a system of natural deduction, uses "introduction" and "elimination" rules. The ones we have seen so far deal with the logical symbol =. The next group of rules deals with the Boolean connectives ∨, ∧ and ¬.

If the conclusion (i.e. the last line) of a Fitch style proof has only one bar on its left side as it should, then it depends only on the premises of the whole argument and not on the various additional assumptions that might appear at the beginnings of other vertical bars.

6.1 Conjunction rules

Conjunction Elimination (∧ Elim)
Removes a conjunct from a previous line containing a conjunction.
This rule allows you to infer any conjunct from a conjunction sentence.

Example:

```
  | 1. Cube(a) ∧ Dodec(c)
  |----
> | 2. Dodec(c)                ∧ Elim: 1
```

Conjunction introduction (∧ Intro)
Introduces a new conjunction on any line of a proof by citing each of the conjuncts from prior lines. These conjuncts must be alone on the line cited. This rule allows one to conjunct any number of previously established statements in any order.

Example:

```
  | 1. Tet(a)
  | 2. Tet(b)
  |----
> | 3. Tet(a) ∧ Tet(b)         ∧ Intro: 1, 2
```

6.2 Disjunction rules

Disjunction introduction (∨ Intro)
The rule allows one to construct any disjunction using previous results as one of its disjuncts.

Example:

```
  | 1.  P
  |----
➢ | 2.  P ∨ Q                         ∨ Intro: 1
```

Disjunction Elimination (∨ Elim)
This is the formal rule that corresponds to the method of proof by cases. It incorporates the formal device of a **subproof**. A subproof involves the temporary use of an additional assumption, which function in a subproof the way the premises do in the main proof under which it is subsumed. We place a subproof within a main proof by introducing a new vertical line, inside the vertical line for the main proof. We begin the subproof with an assumption (any sentence of our choice), and place a new Fitch bar under the assumption:

```
| P ∨ Q
| | P                                 {1st disjunct}
| |----
| | S
|
| | Q                                 {2nd disjunct}
| |----
| | S
| S
```

From the proof above, you need to look at a disjunction that you do have as a premise or an earlier line of the proof, for example, $P_1 \vee \ldots \vee P_n$. Set up a subproof for each disjunct of the disjunction, in this case P_1 to S and then P_n to S . The last line of every one of the subproofs should be the conclusion you are trying to reach. Then the first line after the subproofs will be the conclusion you reached as the last line of each subproof, by Disjunction Elimination.

Solution to exercise 6.3 page 154

1. a = b ∧ b = c ∧ c = d	
2. a = b	∧ Elim: 1
3. b = c	∧ Elim: 1
4. a = c	= Elim: 2, 3
5. c = d	∧ Elim: 1
6. b = d	= Elim: 3, 5
7. a = c ∧ b = d	∧ Intro: 4, 6

Solution to exercise 6.6 page 154

1. (A ∧ B) ∨ (A ∧ C)	
2. A ∧ B	
3. A	∧ Elim: 2
4. B	∧ Elim: 2
5. B ∨ C	∨ Intro: 4
6. A ∧ (B ∨ C)	∧ Intro: 3, 5
7. A ∧ C	
8. A	∧ Elim: 7
9. C	∧ Elim: 7
10. B ∨ C	∨ Intro: 9
11. A ∧ (B ∨ C)	∧ Intro: 8, 10
12. A ∧ (B ∨ C)	∨ Elim: 1, 2-6 and 7-11

6.3 Negation rules

Negation Elimination (¬ Elim)

It means that if you have the double negation, ¬¬P, of a sentence, P, then you are entitled to claim sentence P.

Negation Introduction (¬ Intro)

Assume the opposite of what you want to prove. Derive a contradiction as the last line of the subproof. Get what you want by Negation Introduction as the first line after the subproof. It means that, if you have a subproof starting with sentence P and ending with the contradiction, ⊥, then you are entitled to claim the denial of the sentence you assumed when you started the subproof, i.e., you are entitled to insert ¬P.

⊥ Elimination
It means that, if you have managed to derive the contradiction, ⊥, then you are entitled to insert any arbitrary sentence.

⊥ Introduction
It means that if you have both the sentence P and the sentence ¬P, then you are entitled to claim the contradiction ⊥.

6.4 The proper use of subproofs

It is important to note that when a subproof has ended, none of the lines in that subproof may be cited in any subsequent part of the proof. A subproof begins with an assumption (any sentence of our choice). This can also be called a temporary premise. An assumption is some thought which is introduced into the argument in an effort to reach the conclusion. To distinguish between *premise* and *assumption*, we can say an assumption is used in argument, but is itself 'dropped' when the conclusion is reached. In the end the conclusion is not dependent for its truth on the assumption. When an assumption is made and later dropped we will say that it has been *discharged*.

Never start a subproof unless you know what rule you are using it for, what the assumption should be, what the last line of the subproof should be, and what the first line after the subproof will be.

6.5 Strategy and tactics

A good summary of strategy and tactics is given on page 171 of your prescribed book. Study it.

6.6 Proofs without premises

Both system ***ƒ*** and the program Fitch are set up so that a proof may begin with some line other than a premise. For example, it might begin with a use of **= Intro**. Or, it may begin with a subproof assumption. This means that we may have a proof that has no premises at all! What does such a proof establish? Since a proof establishes that a conclusion is a logical consequence of its premises (i.e., that it must be true if they are), a proof without premises establishes that its conclusion is a logical consequence of the empty set of premises. That is, it establishes that its conclusion must be true, period.

Solution to exercise 6.34 page 175

1. $a = b \wedge Dodec(a) \wedge \neg Dodec(b)$	
2. $a = b$	$\wedge$ Elim: 1
3. $Dodec(a)$	$\wedge$ Elim: 1
4. $Dodec(b)$	= Elim: 2, 3
5. $\neg Dodec(b)$	$\wedge$ Elim: 1
6. $\bot$	$\bot$ Intro: 4, 5
7. $\neg(a = b \wedge Dodec(a) \wedge \neg Dodec(b))$	$\neg$ Intro: 1-6

EXAMPLES TO CONSIDER

Example 1

The task is to prove $\neg P \vee \neg R$ using the premise $\neg(P \wedge R)$.

1. $\neg(P \wedge R)$	
2. $\neg(\neg P \vee \neg R)$	
3. $\neg P$	
4. $\neg P \vee \neg R$	$\vee$ Intro: 3
5. $\bot$	$\bot$ Intro: 4, 2
6. $\neg\neg P$	$\neg$ Intro: 3-5
7. P	$\neg$ Elim: 6
8. $\neg R$	
9. $\neg P \vee \neg R$	$\vee$ Intro: 8
10. $\bot$	$\bot$ Intro: 9, 2
11. $\neg\neg R$	$\neg$ Intro: 8-10
12. R	$\neg$ Elim: 11
13. $P \wedge R$	$\wedge$ Intro: 7, 12
14. $\neg(P \wedge R)$	Reit: 1
15. $\bot$	$\bot$ Intro:13, 14
16. $\neg\neg(\neg P \vee \neg R)$	$\neg$ Intro: 2-15
17. $\neg P \vee \neg R$	$\neg$ Elim: 16

Explanation:

We have to prove ¬P ∨ ¬R using the premise ¬(P ∧ R). This means that one should prove that P is false or R is false whenever it is not the case that P and R are both true.

We start with the premise, i.e. that ¬(P ∧ R) is true, which means that it is not the case that P and R are both true. We are going to use proof by contradiction to show that the conclusion follows from this premise. So, let us assume that the conclusion ¬P ∨ ¬R is false; in other words, let us assume that ¬(¬P ∨ ¬R) is true. In order to reach a contradiction, it is first shown that P must be true, and it is then shown that R must be true. Each of these results is shown by contradiction.

Suppose that P is false. Then ¬P is true and ¬P ∨ ¬R is true as well, contrary to the assumption. That's why P must be true in this case. Secondly, suppose that R is false. Then ¬R is true and ¬P∨¬R is true just like before, contrary to the assumption. For this reason also R must be true.

Thus, P and R are both true so that P∧R is also true. This result contradicts the premise ¬(P ∧ R), however. Thus, it can be concluded that the assumption ¬(¬P ∨ ¬R) must be false and ¬P ∨ ¬R must be true. But this was the conclusion that should have been proved.

Observe that there are steps in the formal proof above which you do not normally write down in an informal proof. For example, when one has deduced a contradiction from ¬P in an informal proof, one normally concludes directly that P is true, instead of first concluding that ¬¬P is true and concluding that P is true only after that. You might wonder how detailed a piece of reasoning must be in order to still count as an informal proof. This is partially a matter of taste, but in a logic course it is expected that proofs are quite detailed. If you are required to give a formal proof in an assignment or examination, the proof should be at least as detailed as the one above.

Example 2

Suppose we have to prove ¬P ∧ ¬Q from the premise ¬ (P ∨ Q). We first give an informal and then a formal proof.

***Informal proof*:**

Let us suppose that the premise ¬ (P ∨ Q) is true. In order to prove ¬P ∧ ¬Q, we first prove ¬P and then we prove ¬Q. Each of these two proofs is a proof by contradiction.

> Assume that P is true. This implies that P ∨ Q is true as well. But this conclusion contradicts the premise that ¬ (P ∨ Q) is true. Thus, the additional assumption P is false and ¬P is true.
>
> Next, assume that Q is true. Also this implies that P ∨ Q is true. But just like before, this conclusion contradicts the premise that ¬ (P ∨ Q) is true. Thus, the additional assumption Q is false and ¬Q is true.

We have concluded that ¬P and ¬Q are both true. Combining these two conclusions one gets ¬P ∧ ¬Q.

Formal proof:

The following formal proof is just another presentation of the argument above.

1. ¬ (P ∨ Q)	
2. P	
3. P ∨ Q	∨ Intro: 2
4. ⊥	⊥ Intro: 3, 1
5. ¬P	¬ Intro: 2 - 4
6. Q	
7. P ∨ Q	∨ Intro: 6
8. ⊥	⊥ Intro: 7, 1
9. ¬Q	¬ Intro: 6 -8
10. ¬P ∧ ¬Q	∧ Intro: 5, 9

Example 3

Suppose we have to prove

(P ∧ Q) ∨ ¬P ∨ ¬Q

without any premises. The formal proof is given on the following page.

1. ¬((P ∧ Q) ∨ ¬(P ∧ Q))	
2. P ∧ Q	
3. (P ∧ Q) ∨ ¬(P ∧ Q)	∨ Intro: 2
4. ⊥	⊥ Intro: 3,1
5. ¬(P ∧ Q)	¬ Intro: 2-4
6. ¬(P ∧ Q)	
7. (P ∧ Q) ∨ ¬(P ∧ Q)	∨ Intro: 6
8. ⊥	⊥ Intro: 7,1
9. ¬¬(P ∧ Q)	¬ Intro: 6-8
10. ⊥	⊥ Intro: 5,9
11. ¬¬((P ∧ Q) ∨ ¬(P ∧ Q))	¬ Intro: 1-10
12. (P ∧ Q) ∨ ¬(P ∧ Q)	¬ Elim: 11
13. P ∧ Q	
14. (P ∧ Q) ∨ ¬P ∨ ¬Q	∨ Intro: 13
15. ¬(P ∧ Q)	
16. ¬(¬P ∨ ¬Q)	
17. ¬P	
18. ¬P ∨ ¬Q	∨ Intro: 17
19. ⊥	⊥ Intro: 18,16
20. ¬¬P	¬ Intro:17-19
21. P	¬ Elim: 20
22. ¬Q	
23. ¬P ∨ ¬Q	∨ Intro: 22
24. ⊥	⊥ Intro: 23,16
25. ¬¬Q	¬ Intro:22-24
26. Q	¬ Elim: 25
27. P ∧ Q	∧ Intro: 21,26
28. ¬(P ∧ Q)	Reit:15
29. ⊥	⊥ Intro: 27,28
30. ¬¬(¬P ∨ ¬Q)	¬ Intro:16-29
31. ¬P ∨ ¬Q	¬ Elim: 30
32. (P ∧ Q) ∨ ¬P ∨ ¬Q	∨ Intro: 31
33. (P ∧ Q) ∨ ¬P ∨ ¬Q	∨ Elim:12,13-14, 15-32

Chapter 7: Conditionals

Outcomes of the chapter

- Understand the construction of truth tables for → and ↔.
- Able to translate from English to FOL using the conditionals.
- Understand conversational implicature.
- Know and able to apply rules for formal proofs involving → and ↔.
- Understand the definition and proof of truth-functional completeness for ∧, ∨, and ¬.

Sections 7.1 – 7.3

These sections introduce you to conditionals and biconditionals. In your textbook they are denoted by → and ↔ respectively. The most important other novelty in these sections is the notion of *conversational implicature*. The distinction between the truth conditions and the conversational implicature of an English sentence should make it easier for you to translate from English into FOL.

As the truth tables of → and ↔ show, these connectives could have been defined in terms of the connectives that were introduced earlier in the book. The conditional P → Q could have been defined as ¬P ∨ Q and the biconditional P ↔ Q could have been defined as (P → Q) ∧ (Q → P) which would, if the above definition is adopted for P → Q, become (¬P ∨ Q) ∧ (¬Q ∨ P).

The *game facility* of Tarski's World makes use of such definitions when it encounters a sentence whose main connective is → or ↔. More precisely, what happens in the game when the considered sentence is of the form P → Q is that the computer replaces the sentence by ¬P ∨ Q but does not change the commitment of the user. For example, if the user has been committed to the truth of P → Q, the computer states, "So you are committed to the truth of ¬P ∨ Q". At this point of the game neither of the players can make any choices between different alternatives. In a similar way, when the main connective of the considered sentence is ↔ so that the sentence is of the form P ↔ Q, the computer just replaces the sentence with (P → Q) ∧ (Q → P).

Maybe it should be emphasised that this does not mean that (for example) P → Q was actually *defined* to be ¬P ∨ Q in your prescribed book. Some logic textbooks do so: they view the sequence of symbols P → Q as an *abbreviation* for ¬P ∨ Q. In a system of logic in which this is the case one is allowed to simply replace each sequence of symbols of the form "P → Q" with the sequence of symbols "¬P ∨ Q". For example, in order to prove "(P → Q) ↔ (¬P ∨ Q)" (which is exercise 8.29 on page 212 of your textbook) in such a system, it would suffice to prove "(¬P ∨ Q) ↔ (¬P ∨ Q)" and to observe that this sentence *is* by definition the same thing as the one that is to be proved. This is *not* legitimate in the system that your textbook uses, however.

Section 7.1 discusses the fact that the conditional $\rightarrow$ does not always have exactly the same meaning as the corresponding English construction "If... then". (No similar problem seems to be associated with the biconditional: only mathematicians and logicians really use its English translation "if and only if", and it does not seem very difficult to accept that it has the meaning given to it by logicians.) An example is the sentence Home(max) $\rightarrow$ Library(claire) which appears on page 179 of your textbook. It does not have exactly the same meaning as *If Max is at home then Claire is at the library*. Of course, the sentences seem to have the same truth value whenever the FOL sentence is false: It seems clear enough that if Max is at home but Claire is not at the library, the English sentence must be false as well. However, as is stated in your textbook, the English sentence can be false also when Max is not at home. Unlike the connective $\rightarrow$, the English expression "if ... then" is not *truth-functional*: the truth value of the English sentence can depend on "other things" besides the truth values of the sentences "Max is at home" and "Claire is at the library".

But what are these other things on which the truth value of the English sentence can depend? According to a standard answer, the English sentence expresses the idea that there is some sort of a connection between Max being at home and Claire being at the library. It has a similar meaning as *If Max were at home, Claire would be at the library.* In other words, the English sentence implicitly contains a notion of *potentiality*; in addition to describing an actual state of affairs it makes a claim which states which non-existing situations are *possible*.

What can logic tell us about such a notion of possibility? The field of philosophical logic which discusses it (among other things) is called *modal logic*. Modal logicians have formulated logical rules that apply to words like "possible" and “necessary". You will learn about this in the third-level logic module that follows on this module. The following four paragraphs are not intended for study but as an appetizer for the third-level module:

> In modal logic, the operators M and N (say) can be put in front of an arbitrary sentence p, and then read as "It is possible that" and "It is necessary that", respectively. These operators can be defined by considering - in addition to the particular situation to which the formal language applies - some set of counterfactual, unreal situations. When such a set of alternative situations has been fixed, one can say that a sentence is *possible* if it is true in at least one of the situations (whether actual or counterfactual), and that a sentence is *necessary* if it is true in all situations.
>
> There are two simple theorems of modal logic that express connections between the operators N and M. Firstly: if p is an arbitrary sentence, then $Np \Leftrightarrow \neg M \neg p$ That is, "p is necessary" is logically equivalent to "It is not possible that p is not the case". Secondly, if p is an arbitrary sentence, then $Mp \Leftrightarrow \neg N \neg p$. In other words, "p is possible" is logically equivalent to "It is not necessary that p is not the case". These connections between N and M are quite analogous with the connections between $\exists$ and $\forall$ that will be discussed in later chapters of your prescribed book.
>
> Using modal logic one can formalise the English sentence "If Max were at home, Claire would be at the library" as N(Home(max) $\rightarrow$ Library(claire)). This can also be viewed as a formalisation of "If Max is at home then Claire is at the library" in so far as the two sentences can be taken to be equivalent. This formalisation states that in all situations in which Max is at home, Claire is at the library.

Of course, the truth value of this formalisation will depend on the set of situations that one uses in the definitions of N and M. More generally, a very essential difference between modal logic and the first-order logic that you are now learning, is that almost all logicians would accept the deduction rules of first-order logic as the correct ones, but there is no single system of modal logic which had found similar general acceptance.

7.4 Truth-functional completeness

It is shown in this section how all possible connectives can be expressed in terms of the three connectives ¬, ∧ and ∨. That is, if * is an arbitrary two-place connective, then there is a sentence which is logically equivalent to P*Q and which has been built from the sentences P and Q using just the connectives ¬, ∧ and ∨.

It is a good exercise to think what would happen if one tried to express these connectives in terms of ¬, ∧ and ∨ in accordance with the instructions that are given on page 191 of your textbook. The answer is that the P → Q would turn out to be equivalent to (P ∧ Q) ∨ (¬P ∧ Q) ∨ (¬P ∧ ¬Q) and P ↔ Q would turn out to be equivalent to (P ∧ Q) ∨ (¬P ∧ ¬Q). Of course, as you may know or remember, P → Q is also equivalent to the much simpler formula ¬P ∨ Q.

One of the questions that might have occurred to you when reading this section is the following: This section tells you how to express any connective in terms of just three connectives; but is it possible to do the same thing with just *two* connectives? The answer is yes: De Morgan's laws imply that the sentence P ∧ Q is equivalent to the sentence ¬(¬P ∨ ¬Q) so that, if one first expresses a connective * using just ¬, ∧, and ∨ and then replaces each part of the resulting sentence which has the form P ∧ Q with a corresponding part of the form ¬(¬P ∨ ¬Q), one gets a sentence which expresses the connective * with just ¬ and ∨. The question which it is natural to ask next is whether one needs at least two connectives in order to be able to express all the other connectives in their terms. Would just *one* be enough? Somewhat surprisingly, the answer is again yes. One connective is enough if one chooses it to be (for example) the connective "neither ...nor". More precisely, let's introduce a new binary connective ■, which is defined by the following truth table:

P	Q	P■Q
TRUE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	TRUE

The sentence P■Q can be read "neither P nor Q" or "not P and not Q", since it is true only when both P and Q are false. Now it is fairly easy to see that all other connectives can be defined using just the connective ■. First it is observed that ¬P can be expressed as P■P. Since P∧Q is clearly equivalent to (¬P)■(¬Q), it is also equivalent to (P■P)■(Q■Q). Further, since P ∨ Q is equivalent to ¬(¬P ∧ ¬Q) and since P■Q is equivalent to ¬P ∧ ¬Q, it immediately follows that P ∨ Q is equivalent to (P■Q)■(P■Q). Finally, since it is shown in this section that any connective can be expressed in terms of ¬, ∧ and ∨, these results imply that it is possible to express any connective in terms of just ■.

Solution to exercise 7.27 page 194

On page 192, the sentence if *P then Q, else R* is translated as

(P ∧ Q ∧ R) ∨ (P ∧ Q ∧ ¬R) ∨ (¬P ∧ Q ∧ R) ∨ (¬P ∧ ¬Q ∧ R).

This can be simplified by using the law of distribution of ∧ over ∨ *in the other direction* that we have done so far. Applying it to the first and the second disjunct on the one hand, and to the third and the fourth disjunct on the other hand, this sentence is seen to be equivalent with

(P ∧ Q ∧ (R ∨ ¬R)) ∨ (¬P ∧ ((Q ∧ R) ∨ (¬Q ∧ R)).

Since (R ∨ ¬R) is a tautology, it does not affect the truth value of the conjunction P ∧ Q ∧ (R ∨ ¬R), and for this reason this conjunction is equivalent with P ∧ Q. On the other hand, by the same distributive law, (Q ∧ R) ∨ (¬Q ∧ R) is equivalent with (Q ∨ ¬Q) ∧ R, which is further equivalent with R since Q ∨ ¬Q is a tautology. These results imply that our formula is equivalent with

(P ∧ Q) ∨ (¬P ∧ R) which is of the required form.

Solution to exercise 7.32 page 196

Of course, the *exclusive disjunction* of two sentences P and Q is true when one, but not when both, of the sentences P and Q are true. That is, P∇Q is true if P is true and Q is false, and if P is false and Q is true, but not otherwise. There are many rules which are valid for exclusive disjunction and which one in principle might wish to add to the system we are using. The following rules seem to be the most obvious choice:

Exclusive Disjunction Introduction (∇ Intro):

```
  | P            | Q
  |              |
  | ....         | ....
  |              |
  | ¬Q           | ¬P
  |              |
  | ....         | ....
  |              |
> | P∇Q      _   | P∇Q
```

Exclusive Disjunction Elimination (∇ Elim):

```
      |   P∇Q
      |
      |  | P ∧ ¬Q
      |  |---
      |  | ....
      |  | S
      |
      |  | ¬P ∧ Q
      |  |---
      |  | ....
      |  | S
>     | S
```

Chapter 8: The Logic of Conditionals

Outcomes of the chapter

- Be able to use the informal methods of proof discussed.
- Be able to use formal rules of proof for → and ↔.
- Understand valid arguments.

8.1 Informal methods of proof

In this section you are introduced to a number of informal proof steps, which include *modus ponens* or conditional elimination, biconditional elimination, contraposition and the method of conditional proof which allows one to introduce a conditional. A rule with which one can introduce biconditionals is not formulated explicitly, although it is made clear on page 202 what this rule is like: since a biconditional P ↔ Q is equivalent to the conjunction of P → Q and Q → P, one is allowed to infer P ↔ Q from P → Q and Q → P. The rules correspond to the formal deduction rules of system ***f*** that are presented in section 8.2: Conditional elimination (→ Elim) corresponds to *modus ponens*, conditional introduction (→ Intro) corresponds to the method of conditional proof, biconditional elimination (↔ Elim) corresponds to the informal rule with the same name, and biconditional introduction (↔ Intro) corresponds to the last of the informal rules mentioned above.

Solution to exercise 8.2 page 203

The file Conditional Sentences.sen contains the following premises:

1. (Small(e) ∧ Dodec(e)) → LeftOf(e,f)
2. Tet(c) → (Tet(e) ∨ Dodec(e))
3. e ≠ c → Tet(e)
4. Small(e) ↔ Dodec(e)
5. LeftOf(e,f) ↔ Larger(f,e)

You should find out, in the case of each of the sentences listed in the exercise on page 204, whether the sentence is a logical consequence of these five premises. If the listed sentence is not a logical consequence, you should make a drawing of a world in which the premises are true but the sentence in question is false. If the sentence is a logical consequence, you should give an informal proof of this fact. We show solutions to the first two listed sentences.

1. The sentence Tet(e) is not a logical consequence of the premises. For example, the conclusion Tet(e) is false in the world which is shown in the figure below, although all the premises are true in it. (We make some cells smaller.) Note that premises 2 and 3 are true in this world because (in both cases) the antecedent is false.

c, **e**: D, S		**f**: D, L					
a: T, S		**b**: C, S		**d**: T S			

2. The sentence Tet(c) → Tet(e) is a logical consequence of the given premises. This is shown by the following informal proof.

Suppose that the given premises are true. That is, suppose that the following sentences are true:

1. (Small(e) ∧ Dodec(e)) → LeftOf(e,f)
2. Tet(c) → (Tet(e) ∨ Dodec(e))
3. e ≠ c → Tet(e)
4. Small(e) ↔ Dodec(e)
5. LeftOf(e,f) ↔ Larger(f,e)

In addition, assume that Tet(c) is true. We want to show that this implies Tet(e) is true as well.

Obviously, if Tet(c) is true, the second premise implies that Tet(e) ∨ Dodec(e). That is, either Tet(e) or Dodec(e) is true. Let us consider these cases separately.

> Suppose first that Tet(e) is true. This is what we are trying to prove at the moment, so in this case we have reached the desired conclusion.
>
> Suppose next that Dodec(e) is true. Now c and e are of a different shape, so that they cannot be two names of the same object. Thus, in this case c ≠ e. But now the third premise implies that Tet(e).

In both cases we have reached the conclusion Tet(e). Thus, together with the premises the additional assumption implies Tet(e), and it can be concluded that the premises imply Tet(c) → Tet(e).

Solution to exercise 8.4 page 204

Here is an informal proof of the argument:

The unicorn, if horned, is elusive and dangerous.
If elusive or mythical, the unicorn is rare.
If a mammal, the unicorn is not rare.

The unicorn, if horned, is not a mammal.

Proof: Assume that the unicorn is horned. We want to prove that it is not a mammal. Assume, by way of a proof by contradiction, that it is a mammal. By the third premise, the unicorn is not rare. By the first premise, the unicorn is elusive. But then by the second premise, the unicorn is rare, after all. This gives us our contradiction, showing us that our assumption is false and we conclude that the unicorn is actually not a mammal.

Solution to exercise 8.7 page 204

Here is an informal proof of the argument:

b is small unless it's a cube.
If c is small, then either d or e is too.
If d is small, then c is not.
If b is a cube, then e is not small.

If c is small, then so is b.

Proof: Assume that c is small. We want to prove that b is small. Assume, by way of contradiction, that b is not small. Then by the first premise, b is a cube. Then by the final premise, e is not small. But then by the second premise, d is small. But the third premise assures us that if d is small, then c isn't small, contradicting our initial assumption. Hence, using proof by contradiction and conditional proof, we see that if c is small, then so is b.

Solution to exercise 8.10 page 205

The file Between Sentences.sen contains the following sentences:

1. Between(a,b,c) $\rightarrow$ Smaller(b,c)
2. Smaller(b,c) $\rightarrow$ [RightOf(b,c) $\wedge$ Between(a,b,b)]
3. Between(c,b,a) $\rightarrow$ [Between(b,a,c) $\vee$ c = b]
4. Between(a,b,c) $\vee$ Between(b,a,c) $\vee$ Between(c,b,a)
5. $\neg$Between(b,a,c)

As stated in this problem, you should either make a drawing of a world in which all these sentences are true, or give a proof which shows that the sentences are contradictory. In the latter case an informal proof will suffice.

It is not possible to draw a world in which all the sentences in the file Between Sentences.sen are true, since these sentences are contradictory. A contradiction can be deduced from them as follows:

Let us assume that all five sentences are true.

- Consider sentence 2. It is not possible for a to be between b and b. For this reason RightOf(b,c) ∧ Between(a,b,b) must be false, and ¬(RightOf(b,c) ∧ Between(a,b,b)) must be true. Now it follows by modus tollens that ¬Smaller(b,c).
- Since Smaller(b,c) is false, sentence 1 implies - again by modus tollens - that ¬Between(a,b,c).
- According to sentence 5, Between(b,a,c) is also false.
- This means that the first two disjuncts in sentence 4 are false so that the third one, Between(c,b,a), must be true.
- But together with sentence 3, Between(c,b,a) implies Between(b,a,c) ∨ c = b. Since it has already been shown that Between(b,a,c) is false, this implies that c = b. But c = b implies ¬Between(c,b,a), since c cannot be between itself and a.

Now we have deduced both Between(c,b,a) and ¬Between(c,b,a), which is a contradiction.

8.2 Formal rules of proof for → and ↔

Rules for conditional

Conditional Elimination

It means that if you have both a conditional, P → Q, and its antecedent, P, then you are entitled to claim its consequent, Q. This rule is often called Modus Ponens.

Conditional Introduction

If you need to prove a conditional, identify the antecedent and consequent of the conditional that you want to prove. Set up a subproof with the antecedent as its assumption, and the consequent as its last line. Then get the conditional as the first step after the subproof, by Conditional Introduction. Make sure the assumption of the subproof is the antecedent of the conditional you are trying to prove, not the antecedent of some conditional you have as a premise or an earlier line.

Look at the diagrams on page 206 of the prescribed book.

Rules for biconditional

Biconditional Elimination

It means that if you have both a biconditional and one of its two sides, then you are entitled to claim the other of the two sides.

Biconditional Introduction

It means that if you have two of the right sorts of sub proofs i.e. one starting with the left side of the biconditional and ending with its right side and the other starting with the right side and ending with the left side, then you are entitled to claim a biconditional.

Look at the diagrams on page 209 of the prescribed book.

8.3 Soundness and completeness

Read for interest, but do not study.

8.4 Valid arguments: some review exercises

Solution to exercise 8.46 page 223

1. Cube(a) $\vee$ (Cube(b) $\rightarrow$ Tet(c))
2. Tet(c) $\rightarrow$ Small(c)
3. (Cube(b) $\rightarrow$ Small(c)) $\rightarrow$ Small(b)

   4. ¬Cube(a)

      5. Cube(a)
      6. $\perp$ — $\perp$ Intro 4, 5
      7. Cube(b) $\rightarrow$ Small(c) — $\perp$ Elim 6

      8. Cube(b) $\rightarrow$ Tet(c)

         9. Cube(b)
         10. Tet(c) — $\rightarrow$ Elim 9, 8
         11. Small(c) — $\rightarrow$ Elim 10, 2

      12. Cube(b) $\rightarrow$ Small(c) — $\rightarrow$ Intro 9-11

   13. Cube(b) $\rightarrow$ Small(c) — $\vee$ Elim 5-7, 8-12, 1
   14. Small(b) — $\rightarrow$ Elim 3, 13

15. ¬Cube(a) $\rightarrow$ Small(b) — $\rightarrow$ Intro 4-14

Solution to exercise 8.48 page 223

Give the formal proof of ¬Tet(c) → FrontOf(a, b) from the following premises:

1. Small(a) ∧ (Medium(b) ∨ Large(c))
2. Medium(b) → FrontOf(a, b)
3. Large(c) → Tet(c)

Here are some hints:

Since the conclusion is a conditional, we will want to use conditional proof strategy (→Intro). So we will start a subproof with ¬Tet(c) as the assumption. But notice that the first premise is a conjunction, and we may need one of its conjuncts separately. (It's always a good idea to use ∧ Elim first, when possible, before setting up any subproofs. That way individual conjunct will be available in all subsequent subproofs.) The second conjunct is a disjunction whose disjuncts occur in the other premises, so first we'll enter it on the next line (∧ Elim) and then set up our conditional proof. The goal of our subproof is FrontOf(a, b), the consequent of the main goal. We can use proof by cases to get it. The first case is easy: Medium(b) gets us our conclusion by → Elim (and the second premise). The second case, namely Large(c), leads to a contradiction, since we get both Tet(c) (from the third premise and → Elim) and ¬Tet(c) which was our assumption. Using ⊥ Elim we derive FrontOf(a, b). That allows us to use ∨ Elim to complete the proof by cases of FrontOf(a, b). Then we use → Intro to complete our conditional proof of ¬Tet(c) → FrontOf(a, b).

Solution to exercise 8.53 page 223

Below is a complete proof using Taut Con.

```
| 1.  Small(a) → Small (b)
| 2.  Small(b) → (SameSize(b, c) → Small(c)
| 3.  ¬Small(a) → (Large(a) ∧ Large(c))
|----
| | 4.  SameSize(b, c)
| |----
| | 5.  Small(a) ∨ ¬Small(a)                   Taut Con
| | | 6.  Small(a)
| | |----
| | | 7.  Large(c) ∨ Small(c)                  Taut Con: 1, 2, 4, 6
| | | 8.  ¬Small(a)
| | |----
| | | 9.  Large(c) ∨ Small(c)                  Taut Con: 3, 8
| | 10. Large(c) ∨ Small(c)                    ∨ Elim: 5, 6-7, 8-9
| 11.  SameSize(b, c) → (Large(c) ∨ Small(c))  →Intro:4-10
```

IMPORTANT LOGICAL EQUIVALENCES AND IMPLICATION RULES

Below we list some equivalences and implication rules.

EQUIVALENCES

1. **DOUBLE NEGATION**

   $A \Leftrightarrow \neg\neg A$

2. **DE MORGAN'S RULES**

   $\neg A \vee \neg B \Leftrightarrow \neg (A \wedge B)$

   $\neg A \wedge \neg B \Leftrightarrow \neg (A \vee B)$

3. **IMPLICATION**

   $A \rightarrow B \Leftrightarrow \neg A \vee B$

4. **CONTRAPOSITION**

   $P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$

5. **BICONDITIONAL EQUIVALENCE**

   $P \leftrightarrow Q \Leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$

6. **DE MORGAN'S LAWS (FOR QUANTIFIERS - see further chapters)**

   $\neg\forall x P(x) \Leftrightarrow \exists x \neg P(x)$

   $\neg\exists x P(x) \Leftrightarrow \forall x \neg P(x)$

IMPLICATION RULES

7. **DISJUNCTIVE SYLLOGISM**

   Given (i) $A \vee B$
   (ii) $\neg A$

   then B follows,

   OR

   Given (i) $A \vee B$
   (ii) $\neg B$

   then A follows

8. **HYPOTHETICAL SYLLOGISM**

Given (i) $A \rightarrow B$
(ii) $B \rightarrow C$

then $A \rightarrow C$ follows

9. **MODUS TOLLENS**

Given (i) $A \rightarrow B$
(ii) $\neg B$

then $\neg A$ follows

10. **MODUS PONENS**

Given (i) $A \rightarrow B$
(ii) A

then B follows

SECTION B: QUANTIFIERS

AIMS AND OBJECTIVES OF SECTION B

The aims of this section:

- to introduce you to quantifiers and their use in first-order languages
- to learn how to translate English sentences to first-order sentences, and vice versa
- to introduce you to methods of proof using the quantifiers

Chapters 9 to 14

These chapters are concerned with FOL formulas which contain quantifiers.

Chapter 9: Introduction to Quantification

Outcomes of the chapter

Understand

- syntax and semantics of quantifiers
- well-formed FOL formulas
- free and bound variables
- the domain of discourse
- Aristotelian forms
- simple translations.

The sections in this chapter introduce you to the quantifiers $\forall$ and $\exists$. The most important concept in these sections is that of a well-formed formula or wff. A well-formed formula of FOL is almost like a sentence in FOL except for the fact that it may contain free variables. As is explained in Section 9.3, a wff is a sentence only when some quantifier binds every occurrence of every variable in it: on page 232 of your textbook a sentence is defined to be a wff with no free variables.

The definition of a wff, which is presented on page 231, is an inductive definition. In general, inductive definitions consist of three parts.

1. An inductive definition begins with a base clause, which describes the basic elements of the set that is being defined. In the case of wffs this description is the statement that an n-ary predicate symbol followed by n variables or individual constants belongs to the set of all wffs. (More specifically, such a wff is called an atomic wff, as your textbook states.)

2. The second part of an inductive definition contains inductive clauses that specify how additional elements of the set are generated from the elements that are already there. In the case of wffs these clauses are listed on page 231-232 of the prescribed book. Each of these clauses tells us how to form a new wff from some given wff or wffs.

3. A rigorous inductive definition would contain, in addition, a final clause stating that there are no other elements of the set than the basic ones and the ones that are generated by the clauses. In the case of wffs the final clause would be the statement that all wffs are either atomic wffs or wffs that can be generated from them by repeatedly applying the seven clauses on page 231-232 of the prescribed book.

A more detailed discussion of inductive definitions is contained in Chapter 16 of the prescribed book, but we are not going to cover that chapter during this course. Should you want to know a bit more about proof by induction, though, keep in mind that the tutorial letter on the mathematical material relevant to you in this course does contain a discussion of the notion of inductive definitions. You will learn there, for instance, that the existence of an inductive definition of a wff is particularly useful when one proves metatheorems about system ***f***.

What precisely do we need the notion of a wff for? Why can we not define sentences directly? The answer is that a direct inductive definition of a sentence of FOL is not possible, because the "building blocks" of sentences of FOL are not always sentences. If we first gave a definition of an atomic sentence of FOL and then presented clauses that were similar to those on page 231-232 of your textbook, but which applied to sentences (for instance, "If P is a sentence, so is ¬P", and so on), the conditions of the resulting definition would not be satisfied by all sentences. For example, ∃x Cube(x) is a sentence, but because Cube(x) is not a sentence, ∃x Cube(x) would not satisfy the conditions of such a definition.

In this chapter there are four Aristotelian forms that are worthy of being studied and it is important to reflect a little on their implications. To help you do this, we give these four forms in terms of some handy logical equivalence below. (You will come across these equivalences as you work through the course, either in the text of your textbook, in the exercises or problems.)

All P's are Q's

Translated into FOL as $\forall x\, (P(x) \rightarrow Q(x))$

$\Leftrightarrow$	$\forall x\, (\neg P(x) \vee Q(x))$	(Implication)
$\Leftrightarrow$	$\forall x\, (\neg\, (P(x) \wedge \neg Q(x)))$	(De Morgan)
$\Leftrightarrow$	$\forall x \neg\, (P(x) \wedge \neg Q(x))$	(Remove redundant brackets)
$\Leftrightarrow$	$\neg \exists x\, (P(x) \wedge \neg Q(x))$	(De Morgan for quantifier)

Some P's are Q's

Translated into FOL as $\exists x\,(P(x) \wedge Q(x))$

$\Leftrightarrow$ $\exists x \neg(\neg P(x) \vee \neg Q(x))$ (De Morgan)

$\Leftrightarrow$ $\neg\forall x\,(P(x) \rightarrow \neg Q(x))$ (Implication & De Morgan for quantifier)

No P is Q

Translated into FOL as $\forall x\,(P(x) \rightarrow \neg Q(x))$

$\Leftrightarrow$ $\forall x\,(\neg P(x) \vee \neg Q(x))$ (Implication)

$\Leftrightarrow$ $\forall x\,\neg\,(P(x) \wedge Q(x))$ (De Morgan)

$\Leftrightarrow$ $\neg\exists x\,(P(x) \wedge Q(x))$ (De Morgan for quantifier)

Some P's are not Q's

Translated into FOL as $\exists x\,(P(x) \wedge \neg Q(x))$

$\Leftrightarrow$ $\exists x\,\neg\,(\neg P(x) \vee Q(x))$ (De Morgan)

$\Leftrightarrow$ $\neg\forall x\,(P(x) \rightarrow Q(x))$ (Implication & De Morgan for quantifier)

Solution to exercise 9.21 page 251

The next table explains the names and the predicates that are used in the translations. (This time function symbols will not be needed.)

ENGLISH	FOL	COMMENTS
Names:		
b	b	The name of the person referred to as "I"
Dee	dee	The name of a river
Denmark	denmark	The name of a country
Logic	logic	The name of a field of study
Predicates:		
x is brave	Brave(x)	
x cares for y	Care(x,y)	
x is a certainty	Certainty(x)	
x is company for y	Company(x,y)	
x deserves y	Deserves(x,y)	
x knows how to forgive	Forgive(x)	
x glitters	Glitters(x)	
x is gold	Gold(x)	
x is a government	Government(x)	

x has y *x is in y* *x is an island* *x is jolly* *x lives near y* *x loves y* *x is a man* *x is a miller* *x is miserable* *x is a nation* *x is a person* *x praises y* *x is a river* *x is rotten* *x is a state*	Has(x,y) In(x,y) Island(x) Jolly(x) Lives(x,y) Loves(x,y) Man(x) Miller(x) Miserable(x) Nation(x) Person(x) Praises(x,y) River(x) Rotten(x) State(x)	

With the help of this table the given sentences can be translated into FOL as follows:

1. ∀x(Forgive(x) → Brave(x))

2. ∀x(Man(x) → ¬Island(x))

This FOL translation captures the *literal* meaning of the English sentence rather than its *metaphoric* meaning which is, of course, what one normally wants to express with a sentence like this.

3. ¬∃xCare(x,b) → ¬∃xCare(b,x)

Notice that in FOL there is nothing that would quite correspond to the personal pronouns of a natural language. Here the name b is being used to denote the person who in this particular sentence is referred to by the pronoun "I".

4. ∀x∀y ((Nation(x) ∧ Government(y) ∧ Has(x,y)) → Deserves(x,y))

5. Certainty(logic) ∧ ∀x((x≠logic) → ¬Certainty(x))

Again, this translation expresses the literal meaning of the English sentence. You might wish to object that it does not make any sense to say that logic *as a whole* is a certainty. Rather, the idea which one normally wants to express with the English sentence is that it is only *logical truths* that are certain. One can express this idea with the FOL sentence ∀x(Logtruth(x) → Certainty(x)) ∧ ∀x(¬Logtruth(x) → ¬Certainty(x)), if one uses Logtruth(x) as the FOL translation of the predicate *x is a logical truth*.

6. If the sentence is taken to mean that *every* miserable person loves *somebody*'s company, its translation will be this: ∀x((Person(x) ∧ Miserable(x)) → ∃y(Company(y,x) ∧ Loves(x,y))

7. This sentence is ambiguous. It could mean that *it is not the case that everything that glitters is gold*, but it could also mean that *it is true of everything that glitters that it is not gold*.

In the former case it should be translated with $\neg\forall x(\text{Glitters}(x) \rightarrow \text{Gold}(x))$ and in the latter case it should be translated with $\forall x(\text{Glitters}(x) \rightarrow \neg\text{Gold}(x))$.

8. $\exists x(\text{Miller}(x) \wedge \text{Jolly}(x) \wedge \text{River}(dee) \wedge \text{Lives}(x,dee))$

Notice that nothing in this translation corresponds to the *past tense* of the corresponding English sentence. If we wanted to express the past tense, we would have to introduce predicates which contained a place for time and which quantified over times.

9. $\forall x((\text{Person}(x) \wedge \forall y(\text{Person}(y) \rightarrow \text{Praises}(x,y))) \rightarrow \forall y(\text{Person}(y) \rightarrow \neg\text{Praises}(x,y)))$

Once more, this is a translation of the literal meaning of the sentence. Note that it implies that there cannot be anyone who would praise everyone.

10. $\text{State}(denmark) \wedge \exists x(\text{Rotten}(x) \wedge \text{In}(x,denmark))$

Chapter 10: The Logic of Quantifiers

Outcomes of the chapter

- Understand
- the truth-functional form algorithm: when are sentences of FOL tautologies?
- the replacement method
- first-order validity and consequence.

This chapter discusses four important and very useful equivalences which involve quantifiers. These are the two De Morgan's laws for quantifiers and the two forms of the Principle of Replacing Bound Variables. It should be observed that De Morgan's laws enable one to define each quantifier in terms of the other one. One could, if one wanted, define the string of symbols $\exists xP(x)$ to be an abbreviation of the string of symbols $\neg\forall x\neg P(x)$. This definition would clearly make sense: if there is some object in the domain of discourse which has the property P, then it is not true that all objects in it fail to have the property P - and this is precisely what $\neg\forall x\neg P(x)$ says. (As a matter of fact, some elementary logic textbooks first consider a language which contains only one quantifier, the quantifier $\forall$, and then introduce the other quantifier $\exists$ with the above definition. However your textbook does not proceed in this way. If it did, the system within which formal proofs are constructed would on the one hand be simpler but on the other hand much less intuitive.)

There are a number of other important quantifier equivalences to be aware of. There are also some important "pseudo-equivalences" to be wary of, i.e. non-equivalences that appear deceptively like equivalences. We list both kinds here.

Distributing $\forall$ through $\wedge$

$$\forall x\,(P(x) \wedge Q(x)) \;\Leftrightarrow\; \forall x\,P(x) \wedge \forall x\,Q(x)$$

Distributing $\exists$ through $\vee$

$$\exists x\,(P(x) \vee Q(x)) \;\Leftrightarrow\; \exists x\,P(x) \vee \exists x\,Q(x)$$

Beware of the following **non**-equivalences:

$$\forall x\,(P(x) \vee Q(x)) \;\Leftrightarrow\; \forall x\,P(x) \vee \forall x\,Q(x)$$
$$\exists x\,(P(x) \wedge Q(x)) \;\Leftrightarrow\; \exists x\,P(x) \wedge \exists x\,Q(x)$$

Notice that you may distribute $\forall$ through $\wedge$, and you may distribute $\exists$ through $\vee$, but you may *not* distribute $\forall$ through $\wedge$ or $\exists$ through $\wedge$. If you are in any doubt about these last two non-equivalences, try problems 10.24 and 10.27. Be sure you understand why the nonequivalent pairs are not equivalent.

Null quantification

In the following examples, P represents any wff in which x does not occur free.

$\forall x\ P \Leftrightarrow P$

$\exists x\ P \Leftrightarrow P$

$\forall x\ (P \vee Q(x)) \Leftrightarrow P \vee \forall x\ Q(x)$

$\exists x\ (P \wedge Q(x)) \Leftrightarrow P \wedge \exists x\ Q(x)$

The last two might be thought of as providing “limited” distribution of $\forall$ through $\vee$ and $\exists$ through $\wedge$. (For an example of the last one, see problem 10.28).

Replacing bound variables

In the next examples, P(x) is any wff and y is any variable that does not occur in P(x):

$\forall x\ P(x) \Leftrightarrow \forall y\ P(y)$

$\exists x\ P(x) \Leftrightarrow \exists y\ P(y)$

What these equivalences tell us, in effect, is that it does not matter which variable we use in a universal or existential generalization. Systematically rewriting the bound variables does not change the meaning of the sentence.

Chapter 11: Multiple Quantifiers

Outcomes of the chapter

- Understand the meaning and use of multiple occurrences of the same quantifier.
- Be aware of translation mistakes: different variables do not mean different objects.
- Understand the meaning and use of mixed quantifiers.
- Able to apply the step-by-step method of translation.
- Be aware of ambiguity and context sensitivity when translating English sentences.
- Understand prenex form and able to transform formulas to this form.

This chapter introduces you to the problem of translating English sentences into FOL sentences that contain quantifiers. Section 11.3 presents a *step-by-step method* of producing such translations, and the next two sections are concerned with applying the method: section 11.4 advises you to reformulate your original sentence before translation whenever this is necessary, and Section 11.5 discusses the translation problems caused by the ambiguity of natural language.

When one translates an English sentence with the step-by-step method, one proceeds *inwards*: one translates the outermost quantifier of the FOL sentence first and leaves within it an expression which contains quite ordinary English words, in addition to FOL symbols. In a second step this expression is then replaced by a wff. The step-by-step method of translation is, basically, the converse of the outwards-proceeding method of translating from FOL to English. Even more than translating from FOL into English, translating from English into FOL is a practical skill which can best be learnt in the doing. In other words the best way to learn is to solve the problems and then to compare your solutions with ours afterwards.

Although Section 11.6 is not prescribed material, please read through it as well as the following three paragraphs: This chapter looks at the versions of FOL that contain function symbols. Everything that can be expressed in a version of FOL which contains function symbols can also be expressed in a version which does not contain them. This chapter gives you an example of how one does this.

One can formulate the sentence "a person's mother is always older than that person" with the function symbol "mother" as

$\forall x[\text{OlderThan}(\text{mother}(x),x)]$

This sentence expresses the idea that each person has precisely one mother, and that she is older than the person. As stated on page 308, the same things can be expressed by the sentence:

$\forall x\exists y[\text{MotherOf}(y,x) \land \text{OlderThan}(y,x) \land \forall z[\text{MotherOf}(z,x) \to y=z]]$

which contains the predicate MotherOf instead of the function symbol mother. The above sentence may look horrendous, but it is not really difficult to understand: Because our universe of discourse (over which the values of variables range) is now the set of all persons, the sentence means that it is true of all persons x that there is some person y who has the following three properties: y is the mother of x, y is older than x, and all mothers of x are identical with y. (The last of these conditions means, in other words, that nobody is a mother of x except for y.) Obviously, this is precisely what the first sentence also said.

A similar translation is acceptable for any function symbol `func` and any predicate `Pred` which corresponds to it. By "corresponding" we mean here that `func` and `Pred` have been *defined* in such a way that, by definition,

`y=func(x)` if and only if `Pred(y,x)`.

In this situation it is easy to express in terms of `Pred` any wff Q which contains `func(x)` and in which one does not quantify over x. (For example, the part of our first sentence which came after "∀x", OlderThan(mother(x),x), was a wff of this kind.)

Section 11.7 presents the rules with which one can transform arbitrary wffs into *prenex normal form*, a form in which all the quantifiers are in front of the wff. If one eliminates the connectives → and ↔ from the wff one is considering by expressing them in terms of ¬, ∧, and ∨, one can always get the wff into a prenex normal form with the rules given on pages 314 and 282.

Solution to exercise 11.37 page 315

The prenex forms of the sentences in the file Jon Russel's Sentences.sen are:

2. Sentence 1 becomes ∀x ∃y [Tet(x) ∨ LeftOf(x, y)]
4. Sentence 3 becomes ∀x ∀y ∃z [(x ≠ y ∧ Cube(x) ∧ Cube(y)) → Between(z, y, x)]
6. Sentence 5 becomes ∀x ∀y[(Cube(x) ∧ Smaller(y, x)) → Medium(x)]
8. Sentence 7 becomes ∃x ∀y [Cube(x) → Small(y)]
10. Sentence 9 becomes ∃x ∀y [Cube(x) → Cube(y)]
12. Sentence 11 becomes ∃x ∃y ∃z [¬Cube(x) ∧ ¬Tet(y) ∧ ¬Dodec(z)]
14. Sentence 13 becomes ∀x ∀y ∃z ¬[Cube(x) ∧ (Tet(y) → Between(z, x, y))]

    Steps: ¬∃x [Cube(x) ∧ ∀y (Tet(y) → ∃z Between(z, x, y))]
    ¬∃x [Cube(x) ∧ ∀y (¬Tet(y) ∨ ∃z Between(z, x, y))]
    ¬∃x [Cube(x) ∧ ∀y ∃z (¬Tet(y) ∨ Between(z, x, y))]
    ∀x¬[Cube(x) ∧ ∀y ∃z (¬Tet(y) ∨ Between(z, x, y))]
    ∀x∀y∃z ¬[Cube(x) ∧ (¬Tet(y) ∨ Between(z, x, y))]
16. Sentence 15 becomes ∃x ∀y ∀z ∃u [(Cube(x) → Small(y)) ∧ (Small(z) → Cube(u))]
18. Sentence 17 becomes ∀x ∃y ∀u ∃v [(Tet(x) ∧ ¬Small(y)) ∨ (Small(u) ∧ ¬Tet(v))]

    Steps: ¬[(∀x Tet(x) → ∀y Small(y)) ∧ (∀y Small(y) → ∀x Tet(x))]
    ¬[(¬∀x Tet(x) ∨ ∀y Small(y)) ∧ (¬∀y Small(y) ∨ ∀x Tet(x))]
    ¬(¬∀x Tet(x) ∨ ∀y Small(y)) ∨ ¬ (¬∀y Small(y) ∨ ∀x Tet(x)) De Morgan
    (∀x Tet(x) ∧ ¬∀y Small(y)) ∨ (∀y Small(y) ∧ ¬∀x Tet(x)) De Morgan, twice
    (∀x Tet(x) ∧ ∃y ¬Small(y)) ∨ (∀y Small(y) ∧ ∃x ¬Tet(x)) De Morgan, twice
    ∀x ∃y ∀u ∃v [(Tet(x) ∧ ¬Small(y)) ∨ (Small(u) ∧ ¬Tet(v))]

20. Sentence 19 becomes

$\forall v\ \forall x\ \exists y\ \forall z\ \exists u\ [(\text{Larger}(x, v) \rightarrow \text{LeftOf}(y, v)) \wedge (\text{LeftOf}(z, v) \rightarrow \text{Larger}(u, v))]$

Steps: $\forall v\ [\neg\exists x\ \text{Larger}(x, v) \vee \exists x\ \text{LeftOf}(x, v)) \wedge (\neg\exists x\ \text{LeftOf}(x, v) \vee \exists x\ \text{Larger}(x, v))]$

$\forall v\ [\forall x\ \neg\text{Larger}(x, v) \vee \exists x\ \text{LeftOf}(x, v)) \wedge (\forall x\ \neg\text{LeftOf}(x, v) \vee \exists x\ \text{Larger}(x, v))]$

$\forall v\ [\forall x\ \neg\text{Larger}(x, v) \vee \exists y\ \text{LeftOf}(y, v)) \wedge (\forall z\ \neg\text{LeftOf}(z, v) \vee \exists u\ \text{Larger}(u, v))]$

$\forall v\ \forall x\ \exists y\ \forall z\ \exists u\ [\neg\text{Larger}(x, v) \vee \text{LeftOf}(y, v)) \wedge (\ \neg\text{LeftOf}(z, v) \vee \text{Larger}(u, v))]$

Solution to exercise 11.45 page 318

If the English predicate *x comes to the party* is translated into FOL with ComesToParty(x) and the predicate *x has to buy more food* is translated with BuyMoreFood(x), and if the constant b is used as the name of the person who is referred to by the word "I" in the two English sentences, their FOL translations will look like this:

1. $\forall x\text{ComesToParty}(x) \rightarrow \text{BuyMoreFood}(b)$
2. $\exists x(\text{ComesToParty}(x) \rightarrow \text{BuyMoreFood}(b))$

According to the first rule which was presented in Problem 33, these sentences are equivalent. However, the English sentences do not seem to be equivalent. This is because the idea which a person who utters the second English sentence,

"*There is someone such that if he comes to the party, I will have to buy more food*",

normally wants to express, is that he or she has to buy more food *for* some particular person if he shows up. If one accordingly introduces a *two-place* predicate BuyMoreFoodFor(x,y), which means

x has to buy more food for y,

and translates the sentences using it, the translations will look like this:

1. $\forall x(\text{ComesToParty}(x) \rightarrow \text{BuyMoreFoodFor}(b,x))$
2. $\exists x(\text{ComesToParty}(x) \rightarrow \text{BuyMoreFood}(b,x))$

Now *there is an occurrence of x in the latter predicate* too and the sentences are no longer equivalent.

How do we proceed when the wff in which we have to pull the quantifiers forward, is in one of the two forms to which the rules do not directly apply? This too is very easy. One just replaces the variable which follows one of the quantifiers with a new one. In this way one can show that

$\forall xP(x) \vee \forall xQ(x) \Leftrightarrow \forall xP(x) \vee \forall yQ(y)$

$\Leftrightarrow \forall x[P(x) \vee \forall yQ(y)] \Leftrightarrow \forall x\forall y[P(x) \vee Q(y)]$

and that

$\exists xP(x) \wedge \exists xQ(x) \Leftrightarrow \exists xP(x) \wedge \exists yQ(y)$

$\Leftrightarrow \exists x[P(x) \wedge \exists yQ(y)] \Leftrightarrow \exists x\exists y[P(x) \wedge Q(y)]$

whenever y is a variable which occurs in neither P(x) nor Q(x).

Chapter 12 and 13:
Methods of Proof for Quantifiers
Formal Proofs and Quantifiers

Outcomes of the chapters

- Understand and be able to apply
- the introduction and elimination rules for ∀ and ∃
- strategies for proofs with quantifiers
- proofs with multiple and mixed quantifiers.

Chapter 12 introduces five informal methods of proof which involve quantifiers. These are universal instantiation, existential generalisation, existential instantiation, the method of general conditional proof, and universal generalisation. The first three and the fifth of these informal methods of proof correspond to the four formal methods of proof which are introduced in Chapter 13, namely Universal Elimination (∀ Elim), Existential Introduction (∃ Intro), Existential Elimination (∃ Elim), and Universal Introduction (∀ Intro), respectively.

Here are a few additional notes on the four rules for quantified sentences in system ***f***

Rules for Quantifiers in system *f*

1 Universal Elimination

```
  | ∀x S(x)
  | .
  | .
  | .
> | S(c)
```

where

- c represents an individual constant - it doesn't matter whether this constant has been used in the proof before or not.
- All variables x occurring in the predicate S(x) must be replaced by the same individual constant.
- We may, of course, use a variable like y instead of an individual constant, but under the condition that y is free in S(y).

```
  | ∀x S(x)
  |  .
  |  .
  |  .
> |  S(y)
```

2 Universal Introduction

```
  |     | [c]
  |     |------
  |     |  .
  |     |  .
  |     |  P(c)
> |  ∀xP(x)
```

where

- c denotes an arbitrary object,
- c does not occur in the main deduction (that is anywhere outside the subproof in which it has been introduced),
- c does not occur in a line obtained by ∃Elim.
- If we use a variable y instead of c, we get

```
  |     | [y]
  |     |------
  |     |  .
  |     |  .
  |     |  .
  |     |  P(y)
> |  ∀xP(x)
```

3 Existential Elimination

```
  |  ∃x S(x)
  |------
  |     | [c]
  |     | S(c)
  |     |------
  |     |  .
  |     |  .
  |     |  .
  |     |  Q
> |  Q
```

where

- c cannot occur in the main deduction (that is anywhere outside the subproof in which it has been introduced).
- We may deduce Q because the same result Q may be concluded from whichever object it is that satisfies S(x).
- Again, we may use a variable instead of c:

$\exists$x S(x)

  [y] S(y)

  .
  .
  .
  Q

> Q

4 Existential Introduction

S(c)
.
.
.
> $\exists$x S(x)

where

- S(x) is the result of replacing c with x.

Hints

1. Consider carefully which kind of quantified formula is involved in a particular sequent.
2. Consider the kind of rule needed to deduce the conclusion.
3. Below we list the steps that you should take for the different combinations of premises and goals.

a) Premises universal and conclusion existential

Apply $\forall$ Elim to all premises
Propositional Logic Rules
$\exists$ Intro

b) Premises and conclusion universal

Assume c for ∀ Intro
Apply ∀ Elim to all premises
Propositional Logic Rules
∀ Intro to discharge c

c) Premises existential and conclusion existential

Assume b and apply ∃ Elim
Propositional Logic Rules
∃ Intro
Quote ∃ Elim to discharge b

d) Premises existential and conclusion universal

c for ∀ Intro
b and ∃ Elim
Propositional Logic Rules
Quote ∃ Elim to discharge b
∀ Intro to discharge c

We see from the above that we need to introduce arbitrary individual constants into our proofs if we want to apply either the rule of ∃Elim or the rule of ∀Intro. Why is it not necessary to do that for ∃Intro or ∀Elim? And why do we need any arbitrary individual constants in the first place?

Consider the following example. Let us take as representation of our domain of discourse the set of all the cars parked in the main car park at Unisa. Let the predicate symbol R(x) denote the fact that x is red.

∀Elim

Say we are told that it is the case that all the cars in the car park are red, that is, ∀xR(x), and we have to find only one red car. Well, any car of all the cars will be red, because we know that all the cars are, indeed, red. So, this is a simple case of picking out one red car from a set containing one or more red cars.

∃Elim

Say we are told that at least one car is red, that is, ∃xR(x), and we have to find one red car. Well, now we know indeed that at least one car is red, but which one? To keep the existentially quantified sentence valid, we have to pick one car at random. The existential sentence implies that at least any one car is red, and if we pick a car about which we have no prior knowledge, we are keeping to that condition. This is why we have to first introduce the arbitrary individual constant before we can eliminate the existential quantifier, but quote the rule of ∃Elim only at the end of the proof. In this way we show that we have indeed selected an arbitrary constant (at least (any) one x) to use in our (sub) proof to eliminate the existential quantifier.

∀Intro

Say we are told there is one red car, that is, R(c). Say we want, actually, in our proof, to deduce a universally quantified sentence from this matter of fact. How can we reason from the fact that we know that one car is red, to the fact that all cars in our universe of discourse are red? Well, the only way is to pick out a car in this domain and to show that this randomly chosen car is indeed red. This rests on the (common sense) fact that any other car could have been picked out as easily and could also have been shown to be red, and so (if we can prove one random car to be red) all the cars in the car park must be red after all. Remember to introduce your arbitrary constant at the beginning of the deduction (subproof) of the universally quantified sentence that you want to get to. This is another illustration of the need to always think about the details of the proof before simply starting to work on it without any clear idea what you are working towards.

∃Intro

Say we are again told that there is one red car, that is, we are told that R(c). Say we now have to prove that there is at least one red car. Well. If we know that one car is indeed red, surely that means that at least one is red? In other words we simply quantify existentially over the sentence R(c) without any further problems.

EXAMPLES ILLUSTRATING THE APPLICATION OF THESE FORMAL PROOF RULES:

1. $R(a,a,a) \vdash \exists z \exists y \exists x\, R(x,y,z)$

	1.	$R(a,a,a)$	
	2.	$\exists x R(x,a,a)$	$\exists$Intro:1 [substitute first a with **x**]
	3.	$\exists y \exists x R(x,y,a)$	$\exists$Intro:2 [substitute second **a** with **y**]
>	4.	$\exists z \exists y \exists x R(x,y,z)$	$\exists$Intro:3 [substitute third **a** with **z**]

Here, **a** is an object from our universe (domain), and that is why we can say that $\exists x R(x,a,a)$.

But, can we deduce $R(a,b) \vdash \exists x R(x,x)$? No, from $R(a,b)$ we can go to $\exists y \exists x R(x,y)$ but we can never substitute the same variable for both **a** and **b**.

2. $\exists x\, (P(x) \wedge Q(x)) \vdash \exists x\, P(x) \wedge \exists x Q(x)$

	1.	$\exists x\, (P(x) \wedge Q(x))$	
	2.	[c] $P(c) \wedge Q(c)$	
	3.	$P(c)$	$\wedge$Elim:2
	4.	$\exists x P(x)$	$\exists$Intro:3 [**c** substituted by **x**]
	5.	$Q(c)$	$\wedge$Elim:2
	6.	$\exists x Q(x)$	$\exists$Intro:5
	7.	$\exists x P(x) \wedge \exists x Q(x)$	$\wedge$Intro:4,6 [note that **x** is not free]
>	8.	$\exists x\, P(x) \wedge \exists x\, Q(x)$	$\exists$Elim: 1, 2–7

3. $\forall x\, R(x,x) \vdash \exists x \exists y\, R(x,y)$

	1.	$\forall x\, R(x,x)$	
	2.	$R(a,a)$	$\forall$Elim:1 [substitute **x** with **a**]
	3.	$\exists y\, R(a,y)$	$\exists$Intro:2 [substitute second **a** with **y**]
>	4.	$\exists x \exists y R(x,y)$	$\exists$Intro:3 [substitute first **a** with **x**]

4. $\exists x \forall y\, R(x,y) \vdash \forall y \exists x R(x,y)$

	1. $\exists x \forall y\, R(x,y)$	
	2. $\boxed{c}$ $\forall y R(c,y)$	
	$\boxed{d}$	
	3. $R(c,d)$	$\forall$Elim:2
	4. $\exists x R(x,d)$	$\exists$Intro:3
	5. $\forall y \exists x R(x,y)$	$\forall$Intro: $\boxed{d}$ – 4
>	6. $\forall y \exists x R(x,y)$	$\exists$Elim: 1, 2–5

5. $\forall x \forall y(R(x,y)\ \ R(y,x)) \vdash R(x,x)$

	1. $\forall x \forall y(R(x,y) \wedge R(y,x))$	
	2. $\forall y\, (R(x,y) \wedge R(y,x))$	$\forall$Elim:1 [**x** substituted by **x**]
	3. $R(x,x) \wedge R(x,x)$	$\forall$Elim:2 [**y** substituted by **x**]
>	4. $R(x,x)$	$\wedge$Elim:3

6 $(\forall x R(x) \wedge \forall x S(x)) \vdash \forall x(R(x) \wedge S(x))$

	1. $\forall x R(x) \wedge \forall x S(x)$	
	2. $\forall x R(x)$	$\wedge$Elim:1
	3. $\forall x S(x)$	$\wedge$Elim:1
	$\boxed{a}$	
	4. $R(a)$	$\forall$Elim:2 [**x** substituted by **a**]
	5. $S(a)$	$\forall$Elim:3 [**x** substituted by **a**]
	6. $R(a) \wedge S(a)$	$\wedge$Intro:4,5
>	7. $\forall x(R(x) \wedge S(x))$	$\forall$Intro: $\boxed{a}$ –6

7. $\exists x(P(x) \vee Q(x)) \vdash \exists xP(x) \vee \exists xQ(x)$

	1. $\exists x(P(x) \vee Q(x))$	
	2. $\boxed{y}$ $P(y) \vee Q(y)$	
	3. $P(y)$	[for $\vee$Elim]
	4. $\exists xP(x)$	$\exists$Intro:3
	5. $\exists xP(x) \vee \exists xQ(x)$	$\vee$Intro:4
	6. $Q(y)$	[for $\vee$Elim]
	7. $\exists xQ(x)$	$\exists$Intro:6
	8. $\exists xP(x) \vee \exists xQ(x)$	$\vee$Intro:7
	9. $\exists xP(x) \vee \exists xQ(x)$	$\vee$Elim:2, 3--5, 6--8
>	10. $\exists xP(x) \vee \exists xQ(x)$	$\exists$Elim:1, 2–9

Chapter 14: More about Quantification

Outcomes of the chapter

- Understanding numerical quantification:
- Be able to express 'there are exactly/at most/at least n things of a certain kind.'

14.1 Numerical quantification

This section is concerned with the presentation of numerical claims in FOL. At the start numerical claims are described as sentences which state that there is a particular number of objects with some particular property. Such sentences have the logical form

there are exactly n objects with the property P

where n is one of the natural numbers 1,2,3, ...

In addition, some of the exercises and problems in this section are concerned with sentences which have one of the following two logical forms:

There are at least n objects with the property P

There are at most n objects with the property P.

It is a good idea to take a closer look at the method of translating some sentences of these forms into FOL. Let us consider a particular natural number n (which might be 1, 2 or 3, and so on) and then start with sentences in the second form. Clearly we need a sentence with at least n variables to say that there are at least n objects with the property P. Let us suppose that we use the variables x_1, x_2, and so on until x_n. (Of course, this means that if n=1 the variable is x_1, if n=2 the variables are x_1 and x_2, if n=3 the variables are x_1, x_2, and x_3, and so on. The strings of symbols below, like '$\forall x_1 \forall x_2 \ldots \forall x_n$' and '$Px_1 \wedge Px_2 \wedge \ldots \wedge Px_n$', should be understood in a similar fashion.)

If we formalise the sentence "x has the property P" as Px, our first attempt at a translation with this choice of variables might be

$\exists x_1 \exists x_2 \ldots \exists x_n [Px_1 \wedge Px_2 \wedge \ldots \wedge Px_n]$.

But this is clearly unacceptable: this "translation" is true even when there is just one object with the property P, since it does not contain the idea that the variables x_1, x_2, and so on up to x_n have *different* values. What we must add to our attempted translation is the statement that all pairs of different variables have different values. To put it schematically,

there are at least n objects with the property P

can be translated into a sentence in the following form:

$\exists x_1 \exists x_2 \ldots \exists x_n [Px_1 \wedge Px_2 \wedge \ldots \wedge Px_n \quad \wedge$ (conjunction of all $x_i \neq x_j$, where i and j are different, between 1 and n, and i<j)].

The condition i<j appears in this schema because it is unnecessary to include both of the equivalent conjuncts $x_i \neq x_j$ and $x_j \neq x_i$ in the conjunction which states that all x_i values are different. For each pair of values of i and j it suffices to include just one of them, and here we have chosen to include the one for which i<j.

The meaning of this schema is illustrated by the following table, which contains the FOL translations of sentences of the form "There are at least n cubes" when n gets the values 1, 2, 3, and 4. (Here we have replaced the variables x_1, x_2, x_3, and x_4 by x, y, z, and u, which are the variables used in the language of Tarski's World.)

ENGLISH	FOL
There is at least one cube.	$\exists x Cube(x)$
There are at least two cubes.	$\exists x \exists y[Cube(x) \wedge Cube(y) \wedge x \neq y]$
There are at least three cubes.	$\exists x \exists y \exists z[Cube(x) \wedge Cube(y) \wedge Cube(z) \wedge x \neq y \wedge x \neq z \wedge y \neq z]$
There are at least four cubes.	$\exists x \exists y \exists z \exists u[Cube(x) \wedge Cube(y) \wedge Cube(z) \wedge Cube(u) \wedge x \neq y \wedge x \neq z \wedge x \neq u \wedge y \neq z \wedge y \neq u \wedge z \neq u]$

Next we consider the last of our three logical forms, the form of sentences which state there are *at most* n objects with some property. We could get acceptable translations for these sentences by simply noticing that they are equivalent to sentences of the form

there are not more than n objects with the property P

and further equivalent to sentences of the form

it is not the case that there are at least n+1 objects with the property P.

We already know how to translate these sentences, since they are negations of the kind of sentences that we considered earlier. However, there is also another way of arriving at a formalisation of the statement that not more than n objects have the property P. One can formalise it by reasoning as follows: Clearly the statement that at most n objects have the property P is equivalent to the statement that if one picks out n+1 different objects they cannot all have the property P — however one happens to choose them. A very first attempt at formalising the second idea might look like this:

$$\forall x_1 \forall x_2 ... \forall x_n \forall x_{n+1} \neg[Px_1 \wedge Px_2 \wedge ... \wedge Px_n \wedge Px_{n+1}].$$

This sentence says that the conjunction of the sentences Px_1, Px_2, ... , and Px_{n+1} is false for all values of the variables x_1, x_2, ... , and x_{n+1}. However, this formalisation is unacceptable since *it does not state that the values of the variables* x_1, x_2, ... , *and* x_{n+1} *are supposed to be different*: for example, the above sentence is also false when there is only one object b with the property P, since if one gives all the variables x_1, x_2, ... , and x_{n+1} the value b in this case, then Px_1, Px_2, ... , and Px_{n+1} will all turn out to be true. What we should have written instead of our attempted formalisation is a sentence which states that, however one gives values to x_1, x_2, ... , and x_{n+1}, if their values are all different, then Px_1, Px_2, ... , and Px_{n+1} are not all true. In other words, we should have said that if Px_1, Px_2, ... , and Px_{n+1} are all true, then some of the variables x_1, x_2, ... , and x_{n+1} have the same value. This means that English sentences of the form we are considering, which is the form:

there are at most n objects with the property P, can be formalised with FOL sentences of the following form:

$\forall x_1 \forall x_2 \ldots \forall x_n \forall x_{n+1}[(Px_1 \wedge Px_2 \wedge \ldots \wedge Px_n \wedge Px_{n+1}) \rightarrow$ (disjunction of all $x_i = x_j$, where i and j are different, between 1 and n+1, and i< j)].

We shall again illustrate our result by presenting a table. The following table contains the FOL translations of the sentences "There are at most n cubes" when n gets the values 1, 2, 3, and 4. Just like before, we will use the Tarski's World version of FOL and replace the variables x_1, x_2, x_3, x_4, and x_5 by x, y, z, u, and v.

ENGLISH	FOL
There is at most one cube.	$\forall x \forall y[(Cube(x) \wedge Cube(y)) \rightarrow x=y]$
There are at most two cubes.	$\forall x \forall y \forall z[(Cube(x) \wedge Cube(y) \wedge Cube(z)) \rightarrow (x=y \vee x=z \vee y=z)]$
There are at most three cubes.	$\forall x \forall y \forall z \forall u[(Cube(x) \wedge Cube(y) \wedge Cube(z) \wedge Cube(u)) \rightarrow (x=y \vee x=z \vee x=u \vee y=z \vee y=u \vee z=u)]$
There are at most four cubes.	$\forall x \forall y \forall z \forall u \forall v[(Cube(x) \wedge Cube(y) \wedge Cube(z) \wedge Cube(u) \wedge Cube(v)) \rightarrow (x=y \vee x=z \vee x=u \vee x= v \vee y=z \vee y=u \vee y=v \wedge z=u \vee z=v \vee u=v)]$

We still have to discuss the claims which state that there is exactly some number n (and not more, and not less) of objects which have some property P. Of course, since such claims are equivalent to statements according to which

there are at least n objects with the property P, and there are at most n objects with the property P

we could translate the claims we are considering now as conjunctions of two of the translations which we can already produce. We can get a somewhat shorter translation by reconsidering our translations of "There are at least n objects with the property P", which were in the form

$\exists x_1 \exists x_2 \ldots \exists x_n[Px_1 \wedge Px_2 \wedge \ldots \wedge Px_n \wedge$ (conjunction of all $x_i \neq x_j$, where i and j are different, between 1 and n, and i<j)].

What this translation says is that it is possible to give the variables x_1, x_2, ... , and x_n values which are all different in such a way that all the sentences Px_1, Px_2, ... , and Px_n turn out to be true. Indeed, the existence of such values clearly shows that there are *at least* n things that have the property P and, obviously, there are *precisely* n things that have the property P if no other things beside these values of

x_1, x_2, ... , and x_n have the property P. This additional condition can be reformulated by saying that if something (call it x_{n+1}) has the property P, then x_{n+1} must be identical to one of x_1, x_2, ... , and x_n.

If we add this condition to our formalisation of the sentences of the form "there are at least n objects with the property P", we get formalisations of

there are exactly n objects with the property P

which in general look like this:

$$\exists x_1 \exists x_2 \ldots \exists x_n [Px_1 \wedge Px_2 \wedge \ldots \wedge Px_n \wedge \text{(conjunction of all } x_i \neq x_j \text{, where i and j are different, between 1 and n, and i<j)} \wedge \forall x_{n+1}[Px_{n+1} \rightarrow (x_{n+1}=x_1 \vee x_{n+1}=x_2 \vee \ldots \vee x_{n+1}=x_n)]].$$

By now it is probably quite obvious to you how one uses this general schema to produce the kind of translations that we have been using as examples. Nevertheless next we shall present a table which contains the translations of four sentences of the form "there are exactly n cubes" into the language of Tarski's World.

ENGLISH	FOL
There is (exactly) one cube.	$\exists x[Cube(x) \wedge \forall y[Cube(y) \rightarrow y=x]]$
There are (exactly) two cubes.	$\exists x \exists y[Cube(x) \wedge Cube(y) \wedge x\neq y \wedge \forall z[Cube(z) \rightarrow (z=x \vee z=y)]]$
There are (exactly) three cubes.	$\exists x \exists y \exists z[Cube(x) \wedge Cube(y) \wedge Cube(z) \wedge x\neq y \wedge x\neq z \wedge y\neq z \wedge \forall u[Cube(u) \rightarrow (u=x \vee u=y \vee u=z)]]$
There are (exactly) four cubes.	$\exists x \exists y \exists z \exists u[Cube(x) \wedge Cube(y) \wedge Cube(z) \wedge Cube(u) \wedge x\neq y \wedge x\neq z \wedge x\neq u \wedge y\neq z \wedge y\neq u \wedge z\neq u \wedge \forall v [Cube(v) \rightarrow (v=x \vee v=y \vee v=z \vee v=u)]]$

Of course, the particular translations given above are not the only FOL sentences that express numerical claims, and sometimes you have to be quite careful in figuring out the precise meaning of an FOL sentence which expresses a numerical claim in some other way.

This section also introduces the uniqueness quantifier $\exists!$ and the numerical quantifiers $\exists^{!n}$. On page 370 of the textbook it is stated that each string of symbols in the form $\exists^{!n}xP(x)$ should be understood as an abbreviation of an FOL sentence which states that there are exactly n objects x which satisfy the condition P(x) but the textbook does not tell us which of the sentences that formalise this idea $\exists^{!n}xP(x)$ corresponds to. The most natural way to interpret the notation $\exists^{!n}xP(x)$ seems to be to see it as an abbreviation of the particular formalisation of "there are exactly n objects which have the property P" which we gave above. The consequence of this is that $\exists^{!1}xP(x)$ and $\exists!xP(x)$ are abbreviations of the same wff except for a possible change of variables, which is caused by the fact that our general schema for formalising the corresponding English sentences did not contain the variable y but the variables x_1, x_2, and so on. Of course, when one applies the schema in a version of FOL where these variables are not used, they should be replaced by some available variables. (In the case of the language of Tarski's World, this means the variables x, y, z, u, v, and w.)

When the wff P(x) is not atomic but has some more complicated logical structure, the definitions of $\exists^{!n}xP(x)$ and $\exists!xP(x)$ should, of course, be understood as follows. For each variable t, by P(t) one means the wff which results from replacing each free occurrence of x in P(x) by t. For example, according to our definition $\exists^{!3}x(Small(x) \leftrightarrow Tet(x))$ is an abbreviation of the following sentence in the language of Tarski's World:

$$\exists x\exists y\exists z[(Small(x) \leftrightarrow Tet(x)) \wedge (Small(y) \leftrightarrow Tet(y)) \wedge (Small(z) \leftrightarrow Tet(z)) \wedge x\neq y \wedge x\neq z \wedge y\neq z \wedge \forall u[(Small(u) \leftrightarrow Tet(u)) \rightarrow (u=x \vee u=y \vee u=z)]].$$

Partial solution to exercise 14.3 page 371

1. $\exists x\exists y\ (Dodec(x) \wedge Dodec(y) \wedge x \neq y)$

4. $\exists x\exists y\exists z\ [x \neq y \wedge x \neq z \wedge y \neq z \wedge \neg Small(x) \wedge \neg Small(y) \wedge \neg Small(z) \wedge \forall v(\neg Small(v) \rightarrow (v = x \vee v = y \vee v = z))]$

Partial solution to exercise 14.6 page 373

1. $\exists x\exists y\ (Tet(x) \wedge Tet(y) \wedge x \neq y)$

4. $\exists x\exists y\exists z\ [x \neq y \wedge x \neq z \wedge y \neq z \wedge Dodec(x) \wedge Dodec(y) \wedge Dodec(z)]$

7. $\forall x(Tet(x) \rightarrow \exists y\ BackOf(x, y))$

10. $\forall x(Dodec(x) \rightarrow (Small(x) \vee Medium(x) \vee Large(x))$

REFERENCES

1. Barwise J and Etchemendy J. (1992) The Languages of First-Order Logic. USA.

2. Copyright © 2004, S. Marc Cohen Revised 11/21/04 Chp1-14.