Notes, Definitions and Comments on
Logic for Prelims
for students in their first year before 2008/2009

NOTES ON HODGES'S LOGIC

extended version

Volker Halbach
New College, Oxford

version of
24th July, 2008

This text is to be used only by candidates sitting Moderations in their second year, that is, Literae Humaniores students, who have studied logic from Hodges's *Logic*. This text is not be used by candidates who are in their first year in 2008/2009.

CONTENT

Preface

In these notes I aim to collect essential definitions and conventions as they are used in (Hodges, 2001). I have tried to stick as closely as possible to this book. I have also included material that goes slightly beyond Hodges' book, but which is usually treated in lectures and classes. In this case I have made an attempt to stick to Hodges' general approach and to follow widely accepted conventions.

In the margin I have given numbers such has 'H22' that refer to the corresponding page in (Hodges, 2001).

I am indebted to Stephen Blamey, Dave Leal and Hugh Rice for explaining to me Hodges' terminology and for suggesting numerous clarifications and improvements. I also thank Stephen Blamey for making the tableaux proofs more bearable by helping me with the macros.

Sebastian Sequoiah-Grayson (and some others) spotted several mistakes. I thank them for the corrections.

This pdf file contains internal links. A click on text in a red or green frame will send you to the corresponding definition or reference. These features get lost when the Manual is printed on paper.

Text typeset in grey does not form part of the official Logic Manual. This text is only found in the extended version of the Logic Manual. Usually the additional text provides only examples, further explanations and personal remarks.

Motivation

> Praktischen Nutzen wird die *Logik*, wenigstens für das eigene Denken, nicht leicht haben.
>
> Arthur Schopenhauer, *Die Welt als Wille und Vorstellung* II, first book, chapter 9

The following remarks are quite idiosyncratic. I could not resist the temptation to use the opportunity for some ideological comments. They do not contribute much to the elucidation of the Manual, so they may be skipped without any detrimental effect for the understanding of the Manual. Actually this section might only become understandable once one has mastered the remaining content of the Manual.

One reason to write this section was that I am uncomfortable with the introductory remarks in many elementary logic books. In many of them one can read that the study of logic helps to clarify one's thinking. While this may be true to some extent, I do not think that the main point of teaching logic is to teach you how to think or how to think in a more lucid way. Learning logic will teach you how to think as little as learning the physical theory of bicycling will teach you how to ride a bicycle. The theory may help you in some extraordinary situations, which are admittedly numerous in philosophy, but the real attraction of logic lies elsewhere.

Here I shall not try to say what logic is. Rather I shall sketch some reasons for thinking that logic might be important in philosophy.

If one tries to make general statements about natural languages like English, one is confronted with the bewildering complexity of these languages. Natural languages offer a enormous variety of stylistic variants for expressing something. The style in which some argument is presented may help the understanding, but the validity of an argument does not depend on the style of its presentation. Therefore since antiquity logicians have tried to abstract from the particular style in which an argument is presented and to regiment the style of presentation. This regimentation of style has been carried forward until the natural languages under consideration had become purely formal artificial languages. Of course, it still hoped that English sentences can be translated into sentences of these formal languages, but on the loss of the stylistic richness of the English language.

The sheer complexity of the grammar of natural languages drives linguists to despair. Text processing programmes can check the spelling, but they still come without a grammar checker. In contrast, the description of the grammar of the formal language $\mathscr{L}_2$ in section 3.1 below fits on a single page, although expressively rich enough to carry out all kinds of mathematics in it. Of course, $\mathscr{L}_2$ is not as flexible, concise, subtle, potentially elegant and rich in stylistic variants as natural languages, but $\mathscr{L}_2$ has not been devised

to be used in the same as natural languages are used. One can view $\mathscr{L}_2$ as English stripped from all stylistic adornment and ambiguities to the bare bones that are relevant for reasoning. Whether some essential feature has been stripped away together with the adornment, is a matter of dispute, but even if something had been lost, another formal language could be proposed.

Once language has been simplified and regimented, certain of its aspects can be scrutinised that are much less accessible without regimentation and simplification. Here I shall give only some examples.

First example. In order to show that a conclusion follows from a certain theory, one will usually try to give an argument for the conclusion from the assumptions in the theory. Some sort of regimentation may be useful, but it comes in its own when it comes to show that the conclusion does *not* follow from he theory. It will hardly be possible to browse through all potential arguments and check whether they really arguments for the conclusion from the assumptions of the theory. Logicians have devised many ingenious methods for showing that certain sentences do not follow from certain assumptions. This is especially important when neither the sentence nor its negation are consequences of the theory, that is, when the sentence is independent from the theory. A mathematician, for instance, may first try to prove a hunch from certain assumptions. Then he gets stuck and starts to doubt his first hunch. Thus he starts to disprove his hunch by proving its negation. If the sentence is independent from the theory, neither strategy can be successful. Actually some problems in mathematics turned out to be of this kind and logicians were able to show that certain conjectures were simply not decidable on the basis of the available assumptions.

A very special kind of conclusion are contradictions. If a sentence and its negation can be derived from a theory, the theory called 'inconsistent'. Obviously, its bad for a theory to be inconsistent, so often people try to show that their theory is safe from contradiction. Again they may avail themselves of the tools of logic.

Second example. Philosophers have tried to provide semantics for language. They have tried to develop theories about the meanings of sentences and certain words. This may sound like a problem for linguists, but we are also at the core of metaphysics. For some philosophers have claimed that abstract or universal objects like properties are required as meanings. Others, the nominalists, have denied the need for such objects.

One can try the semantical theories first on the highly regimented languages of logic: any semantical theory that does not work for these simply languages will also fail to work for the more complex natural languages.

The semantics of formal languages belongs to logic itself. In the Manual only the semantics of propositional logic will be covered in section 2.2. I consider it a major shortcoming of this course that semantics for predicate

logic is not included, because this is where things become really interesting.

Semantics usually also yields a theory of truth. Logicians have developed precise formal semantics for formal languages including precise theories of truth. Many philosophers have been so overwhelmed by these discoveries that they thought that a solution to the age-old problem of truth has been found.

Third example. So far it may seem that logicians have done everything to chop down language into pieces in order to trivialise language and reasoning. But logic has helped to bring out those complexities of language that cannot be eliminated.

1 Preliminaries

1.1 Sets[†]

A set is a collection of arbitrary objects. Sets are identical if and only if they have the same members. Therefore the set of all animals having kidneys and the set of all animals having a heart are identical, because exactly those animals that have kidneys also have a heart and vice versa.[1] In contrast, the property of having a heart is distinguished from he property of having kidneys. Here we are only interested in sets, but not in properties.

The objects belonging to a set are its elements. $a \in M$ expresses that a is an element of the set M. If a is an element of M, one also says that a is in M.

There is only one set that contains no elements, namely the empty set $\emptyset$.

There are various ways to denote special sets. The set {New College, Merton College} has exactly the two colleges as its elements. The set {Merton College, New College} has the same elements. Therefore the sets are identical, that is, we have:

$$\{\text{New College, Merton College}\} = \{\text{Merton College, New College}\}$$

Thus if a set is specified by including names for the elements in curly brackets, the order of the names does not matter. The set {New College, Merton College, St. Mary of Winchester College} is again the same set because 'St. Mary of Winchester' is just another name for New College, and the set has therefore the same elements as {New College, Merton College}.

Above I have been talking about the set of all animals with a heart. This can be written more formally as:[2]

$$\{x : x \text{ is an animal with a heart}\}$$

[†] Even readers who dislike mathematics may not omit this section.

[1] I have added this footnote because there are regularly protests with respect to this example. For the example only complete and healthy animals are considered. I was told that planarians are an exception, so we would have to exclude them for the sake of the example.

[2] One must exercise some caution here. Certain natural assumption on the existence of sets, lead to the mathematician's hell: inconsistency. This footnote is for those who have heard about Russell's paradox; those who haven't shouldn't worry and ignore this footnote.

1.2 Ordered pairs and relations

As pointed out above, the elements of a set are not ordered by the set. {Tony Blair, George W. Bush} is the same set as´{George W. Bush, Tony Blair}. Ordered pairs, in contrast, have an order on their components. One writes ⟨George W. Bush, Tony Blair ⟩ for the ordered pair with George W. Bush as the first and Tony Blair as the second component. ⟨George W. Bush, Tony Blair ⟩ and ⟨Tony Blair, George W. Bush⟩ are different ordered pairs, because the former has George W. Bush as first component, the latter Tony Blair (the second components are also different).[3]

A set is a binary relation if and only if it contains only ordered pairs.[4] In particular, the empty set Ø is a relation, because it does not contain anything but ordered pairs.

The relation that is satisfied by objects x and y if and only if x is smaller than y is the following set:

> {⟨Munich, London ⟩, ⟨ Oxford, London⟩, ⟨Oxford, Munich⟩, ⟨Munich, Paris⟩,. . . }

Here and in the following I'll use 'iff' as an abbreviation for 'if and only if'.

In the following definition $\mathscr{D}$ is an arbitrary set. $\mathscr{D}$ may be empty.

Definition 1.1. *A relation R is* H146

(i) *reflexive on $\mathscr{D}$ iff for all a in $\mathscr{D}$ $\langle a,a\rangle \in R$.*

(ii) *irreflexive iff for no a $\langle a,a\rangle \in R$.*

(iii) *non-reflexive iff R is neither reflexive nor irreflexive.* H147

(iv) *symmetric iff for all a,b: if $\langle a,b\rangle \in R$ then $\langle b,a\rangle \in R$.*

(v) *asymmetric iff for no a,b: $\langle a,b\rangle \in R$ and $\langle b,a\rangle \in R$.*

(vi) *non-symmetric iff R is neither symmetric nor asymmetric.* H149

(vii) *transitive iff for all a,b,c: if $\langle a,b\rangle \in R$ and $\langle b,c\rangle \in R$, then $\langle a,c\rangle \in R$.*

(viii) *intransitive iff for all a,b,c: if $\langle a,b\rangle \in R$ and $\langle b,c\rangle \in R$, then not $\langle a,c\rangle \in R$.*

(ix) *non-transitive iff R is neither transitive nor intransitive.* H150

(x) *connected on $\mathscr{D}$ iff for all a,b in $\mathscr{D}$ either $\langle a,b\rangle \in R$ or $\langle b,a\rangle \in R$ or $a = b$.*

(xi) an equivalence relation on $\mathcal{D}$ iff it is reflexive on $\mathcal{D}$, symmetric and transitive.

Note. Hodges (2001) does not refer explicitly to the domain in his definition of reflexivity. If the domain is empty, for instance, the empty relation Ø is reflexive; if the domain is not empty, then the empty relation Ø is *not* reflexive, because by the definition on p. 146 *every* dot must have a loop. Here I have made the reference to the underlying set explicit. The set $\mathcal{D}$ is Hodges' domain. The definition of connectedness suffers from a similar problem.

Definition 1.2. *A relation is a* function *iff for all a,b,c: if $\langle a,b\rangle \in R$ and $\langle a,c\rangle \in R$ then $b = c$.*

For instance the relation that is satisfied by those pairs $\langle a,b\rangle$ such that b is the mother of a is a function, because nobody has two (different) mothers.

There are also three-place relations. These are sets of triples $\langle a,b,c\rangle$. Four-place relations are sets of quadruples and so on. Unary (one-place) relations are simply sets.

1.3 Arguments, validity and contradiction

In logic usually sentences are the objects that can be true or false. Of course not every sentence of English can be true: a question like 'Are there any crisps with fish flavour?' is neither true nor false. The following focuses on declarative sentences, that is, sentences that can be true or false.

A sentence can be true in one possible situation and false in others.

An argument consists in premisses and one conclusion. Premisses and conclusion are declarative sentences. The following is an example of an argument:

> There is no German who does not like crisps with paprika flavour.
> Rebecca and Johannes are German.
> *Therefore* Johannes likes crisps with paprika flavour.

The two sentences 'There is no German who does not like crisps with paprika flavour.' and 'Rebecca and Johannes are German.' are the premisses of the argument, while 'Johannes likes crisps with paprika flavour.' is its conclusion.

[3] Using a nice trick one can dispense with ordered pairs. One can define the ordered pair $\langle a,b\rangle$ as $\{\{a\},\{a,b\}\}$. I don't show here that this definition does the trick. The trick will not be used here.

[4] There are also other notions of relations. According to another use of the term binary relations relate to sets of ordered pairs in the same way as properties are related to sets. Here a relation is identified with the corresponding set of ordered pairs. This is common practice in mathematics.

Usually the conclusion is marked by a phrase like 'therefore' or 'it follows that', but it need not be. Sometimes the conclusion precedes the premisses:

> Johannes likes crisps with paprika flavour. *For* there is no German who does not like crisps with paprika flavour, and Rebecca and Johannes are German.

An argument may feature just one premiss or, as a degenerate case, no premiss at all. H38

An argument is valid if and only if there is no possible situation in which all the premisses of the argument are true and the conclusion is false. The arguments above are valid.

An argument is invalid if and only if there is at least one possible situation where all the premisses are true and the conclusion is false.

An argument with no premisses will be valid if and only if the conclusion is true in all possible situations. A sentence is a necessary truth if and only if it is true in all possible situations.

A sentence is consistent if and only if it is true in at least one possible situation. A set of sentences is consistent if there is at least one possible situation where all its elements are true. H1

A sentence is a contradiction, inconsistent or self-contradictory if and only if there is no possible situation where it is true. A set of sentences is inconsistent if and only if there is no possible situation in which all its elements are true.

A sentence is contingent if and only if it is true in at least one possible situation and false in at least one possible situation.

1.4 Syntax, semantics, and pragmatics

In the following formal languages will be studied. They are in many respects much less complicated that natural languages like English or German, but they are intended to mirror certain properties of natural languages. Some philosophers conceive these formal languages as models for natural languages.

Traditionally three aspects in the analysis of languages are distinguished: syntax, semantics and pragmatics. In order to use a language competently, one must master all three aspects of a language.

Syntax is exclusively concerned with the expressions of a language bare of any meaning. For instance, in the syntax of a language it is specified which expressions are words or sentences of the language. In general, grammar belongs to the syntax of a language (and often syntax is identified with grammar). In order to use the language competently, one must know the grammar of the language. In particular, one must know how to form sentences in the

language.

Semantics may be described as the study of the meaning of the expressions of a language. It is of little use to know the grammar of a language without knowing anything about the meaning of the words and sentences of a language.

A competent speaker of German will realise that 'Im Mondschein hockt auf den Gräbern eine wild gespenstische Gestalt.' is a well formed German declarative sentence. In a syntactic analysis of that sentence one may remark that 'den Gräbern' is in the dative case, 'hockt' is a verb in present tense and the singular and so on. All this does not tell you anything about the meaning of that sentence. In order to understand the sentence, you need information about meaning. For instance, it is a semantic fact of German that 'Gräbern' refers to graves, and 'hockt' means 'crouches'.

Syntax and semantics are intertwined and the distinction is not completely sharp. In the following the syntax and the semantics of the formal languages will be introduced. First I shall exhibit the grammar of a language and then specify the meaning of (some of) the expressions of the language.

The third component, pragmatics, will not be studied. Pragmatics is, roughly speaking, the study of language in use. Assume John calls Mary and asks her to come along to the cinema. Now she replies 'I am ill.'. Obviously, John should not expect Mary to come along, but that information is not contained in the meaning of the sentence 'I am ill.'; this sentence says merely something on Mary's health. The sentence 'I am ill.' does not imply that I am not going to the cinema. But this sentence spoken by Mary in this situation conveys the information that she will not join John. Thus John needs pragmatics in order to understand that Mary is not coming along. Pure semantics would not tell him that he is not coming along.

2 Propositional logic

2.1 The syntax of the language of propositional logic

H97

This exposition deviates from section 26 in (Hodges, 2001) in taking 'P', 'Q' and so on as sentence letters right from the beginning instead of 'P_0', 'P_{00}',... Hodges says later on that he uses 'P', 'Q',... instead of 'P_0', 'P_{00}',... [1].

Definition 2.1 (sentence letters)**.** *'P', 'Q', 'R', 'S', 'T', 'P_1', 'Q_1', 'R_1', 'S_1', 'T_1', 'P_2' and so on are sentence letters (or, as some authors say, propositional variables, parameters or constants).*

Definition 2.2 (formula of $\mathscr{L}_1$)**.**

(i) All sentence letters are formulae of $\mathscr{L}_1$.

(ii) If ϕ and ψ are formulae of $\mathscr{L}_1$, then '$\neg\phi$', '$[\phi \wedge \psi]$', '$[\phi \vee \psi]$', '$[\phi \rightarrow \psi]$' and '$[\phi \leftrightarrow \psi]$' are formulae of $\mathscr{L}_1$.

Of course, nothing else is a formula. This could be added explicitly as a further condition in the definition, but I introduce the convention that the definition above and in the following have to be understood in the most restrictive way. That is, I ask the reader to add the clause 'and nothing else is a formula of $\mathscr{L}_1$.'

The Greek letters 'ϕ' and 'ψ' in (ii) are not expressions of the language $\mathscr{L}_1$. For instance, '$[\phi \wedge \psi]$' is not a formula of $\mathscr{L}_1$ and it only becomes a formula of $\mathscr{L}_1$ if 'ϕ' and 'ψ' are replaced by formulae of $\mathscr{L}_1$, respectively.

In Definition 2.1 it would not have been sufficient to say that '$[P \wedge Q]$' etc. is a formula. (ii) rather implies that '$[\phi \wedge \psi]$' is a formula even if 'ϕ' and 'ψ' are replaced by very long formulae.

I could have formulated $\mathscr{L}_1$ without using the 'metavariables' 'ϕ' and 'ψ' on the cost of readability by expressing (ii) in the following way.

> The negation symbol followed by a formula of $\mathscr{L}_1$ is again a formula of $\mathscr{L}_1$. The symbol '[' followed by a formula of $\mathscr{L}_1$, followed by the symbol '$\vee$' (or '$\wedge$', '$\rightarrow$', '$\leftrightarrow$'), followed by a formula (not necessarily distinct from the first one), followed by the symbol ']' is a formula of $\mathscr{L}_1$.

[1] For the use of quotation marks see Appendix C

I hope that (ii) is not only shorter but also easier to grasp.

Example 2.3. By (i), 'P' is a formula of $\mathscr{L}_1$. Thus by (ii) '$\neg P$' is also a formula of $\mathscr{L}_1$. By (i) again 'T_4' is a formula of $\mathscr{L}_1$. By (ii) and what has been said so far, '$[\neg P \wedge T_4]$' is a formula, and by (ii) again also '$[[\neg P \wedge T_4] \rightarrow P]$' is a formula of $\mathscr{L}_1$.

Hodges (2001) calls the symbols '$\neg$', '$\wedge$', '$\vee$', '$\rightarrow$', '$\leftrightarrow$' truth-functor symbols. Most other authors call them connectives.

name	in English	symbol	alternative symbols
conjunction	and	$\wedge$	., &
disjunction	or	$\vee$	+
negation	it is not the case that	$\neg$	-, ~
arrow (material implication, conditional)	if—then—	$\rightarrow$	$\supset$
biconditional (material equivalence)	if and only if	$\leftrightarrow$	$\equiv$

The names in brackets and the symbols in the right column are used by other authors; they will not be used in exams. But it is useful to know them when you are reading other authors.

The expressions in the 'in English' column indicate how the connectives are commonly read, rather than their precise meanings.

2.2 The semantics of propositional logic

> We could forget about philosophy. Settle down and maybe get into semantics.
>
> Woody Allen Mr. Big, in *Complete Prose*, Basingstoke: Picador, 1997, 283–292

H99ff

Definition 2.4. *A $\mathscr{L}_1$-structure is a function that assigns a truth value (T or F) to some sentence letters.*

Definition 2.5. *Let* **A** *be some structure that assigns either T or F to every sentence letter in ϕ and ψ.*

(i) If **A** *assigns T to a sentence letter, then the sentence letter is true in* **A**.

(ii) If ϕ is not true in **A**, *then '$\neg\phi$' is true in* **A**.

(iii) If ϕ and ψ are true in **A**, *then '$[\phi \wedge \psi]$' is true in* **A**.

(iv) If ϕ or ψ (or both) is true in **A***, then '$[\phi \vee \psi]$' is true in* **A***.*

(v) If ϕ is not true in **A** *or ψ is true in* **A***, then '$[\phi \rightarrow \psi]$' is true in* **A***.*

(vi) If ϕ and ψ are both true or if ϕ and ψ are both false, then '$[\phi \leftrightarrow \psi]$' is true in **A***.*

(vii) If ϕ is not true in **A***, then ϕ is false.*

The assumption that **A** assigns T or F to all sentence letters in ϕ is important. If this condition is not satisfied, ϕ is neither true nor false in **A**.[2]

The definition can be summarised in truth tables. These tables allow one to calculate whether a sentence is true or false in a structure. For instance, the first line of the table for $\wedge$ tells you that '$[\phi \wedge \psi]$' is true in the structure iff ϕ is true and ψ is true in this structure.

ϕ	$\neg\phi$
T	F
F	T

ϕ	ψ	$[\phi \wedge \psi]$
T	T	T
T	F	F
F	T	F
F	F	F

ϕ	ψ	$[\phi \vee \psi]$
T	T	T
T	F	T
F	T	T
F	F	F

ϕ	ψ	$[\phi \rightarrow \psi]$
T	T	T
T	F	F
F	T	T
F	F	T

ϕ	ψ	$[\phi \leftrightarrow \psi]$
T	T	T
T	F	F
F	T	F
F	F	T

Definition 2.5 determines whether a formula is true or false in a structure, if the structure assigns T or F to every sentence letter in the sentence. I explain this by an example.

A structure **A** assigns T to the sentence letter 'P' and F to the sentence letter 'Q'; and it is to be determined whether the formula '$[\neg[P \rightarrow Q] \rightarrow [P \wedge Q]]$' is true in this structure or not.

Since **A** assigns T to 'P', 'P' is true in **A**; and since **A** assigns F to 'Q', 'Q' is false in **A** by Definition 2.5 (i). By Definition 2.5 (v), '$[P \rightarrow Q]$' is false in **A** and thus '$\neg[P \rightarrow Q]$' is true in **A** by Definition 2.5 (ii). By Definition 2.5 (ii) the formula '$[P \wedge Q]$' is false in **A**, and therefore the entire formula '$[\neg[P \rightarrow Q] \rightarrow [P \wedge Q]]$' is false in **A** according to Definition 2.5 (v) again.

[2] Here Hodges deviates again from the usual definition. The definition would become smoother if it were assumed that the structure assigns truth values to *every* sentence letter.

This way of writing down the calculation is awkward. It becomes much more perspicuous if written down in the following way:

P	Q	$[\neg$	$[P$	$\rightarrow$	$Q]$	$\rightarrow$	$[P$	$\wedge$	$Q]]$
T	F	T	T	F	F	**F**	T	F	F

The boldface F is the final value.

One can calculate the truth and falsehood of a formula for all possible structures that assign T or F to all sentence letters of the formula in a single truth table:

P	Q	$[\neg$	$[P$	$\rightarrow$	$Q]$	$\rightarrow$	$[P$	$\wedge$	$Q]]$
T	T	F	T	T	T	**T**	T	T	T
T	F	T	T	F	F	**F**	T	F	F
F	T	F	F	T	T	**T**	F	F	T
F	F	F	F	T	F	**T**	F	F	F

Again the column that indicates the truth and falsehood of the entire formula is in boldface letters. In the following definition I call this column the main column.

Definition 2.6.

- *A formula is a tautology if and only if there are only Ts in the main column of its truth table.*
- *A formula is semantically inconsistent if and only if there are only Fs in the main column of its truth table.*
- *A formula is a propositionally contingent if and only if there are Ts and Fs in the main column of its truth table.*

Some authors call tautologies 'logically necessary' or 'valid' formulae.

Definition 2.7. *Let X be a finite set of formulae of $\mathscr{L}_1$ and ψ be a formula of $\mathscr{L}_1$.*

(i) *$X \vDash \phi$ iff there is no structure such that all formulae in X are true in the structure and ϕ is false in it.*

(ii) *$X \vDash$ iff there is no structure such that all formulae in X are true in it.*

Hodges calls expressions of the form '$X \vDash$' and '$X \vDash \phi$' 'semantic sequents'. These formal expressions are read in English in the following way:

Definition 2.8. *(i) A set X of formulae is (semantically) inconsistent iff $X \vDash$.* H102

(ii) An inference from the formulae in X to ϕ is valid *iff $X \vDash \phi$.*

(iii) X semantically entails ϕ iff $X \vDash \phi$.

More precisely, one should talk about inconsistency etc. *in propositional logic*. When there is a danger of confusion this specification should be added.

Lemma 2.9. *$X \vDash \phi$ iff $X, \neg\phi \vDash$.*

Here $X, \neg\phi$ is the set with ϕ and all elements of X as elements.

Proof. Assume $X \vDash \phi$. Then there is no structure **A** such that all formulae in X are true in the structure and ϕ is false in **A**. Therefore there is no structure such that all formulae in X are true in the structure and '$\neg\phi$' is true in **A**. Hence there is no structure that assigns T or F to all sentence letters in $X, \neg\phi$ such that all formulae in $X, \neg\phi$ are true in **A**.

Assume $X, \neg\phi \vDash$. Then there is no structure **A** that assigns T or F to all sentence letters in $X, \neg\phi$ such that all formulae in $X, \neg\phi$ are true in **A**. And thus there is no structure that assigns T or F to all sentence letters in $X, \neg\phi$ such that all formulae in X are true in **A** and ϕ is false in **A**; and therefore $X \vDash \phi$.

We have only truth functor symbols for certain truth functors. That is, we do not have for any possible truth table a corresponding symbol. For instance, we do not have a truth functor symbol for the exclusive 'or' with the following truth table:

ϕ	ψ	$[\phi \dot{\vee} \psi]$
T	T	F
T	F	T
F	T	T
F	F	F

The addition of a new symbol is not really necessary, because the $\dot{\vee}$ can be defined with those we already have: $[P \dot{\vee} Q]$ has he same truth table as $[[P \vee Q] \wedge \neg[P \wedge Q]]$ or as $\neg[P \leftrightarrow Q]$:

P	Q	$\neg$	$[P$	$\leftrightarrow$	$Q]$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	T	F	F	T
F	F	F	F	T	F

By providing suitable formulae for every truth table with two sentence letters one can prove that there is no need to add a further binary truth functor symbol. The task is much harder for truth tables with three sentence letters, because there are much more cases to consider. In order to prove the claim for arbitrary truth tables, a general procedure for specifying the formula is required. This can be done, but I skip the proof.

2.3 Proofs

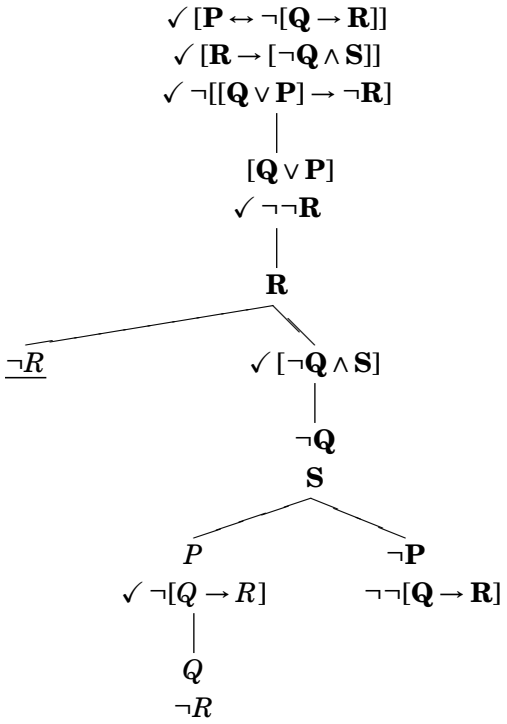
> Don't eliminate cut.
> George Boolos, 'Don't eliminate cut' in *Logic, Logic, and Logic*

H115ff

A tree (of formula of $\mathscr{L}_1$) has some formula at the top and it is branching to the bottom (it is not allowed that the branches of the tree merge again).

A branch in a tree is a sequence of formulae with a formula on the bottom as the last element and the topmost formula as the first element and all formulae in between (in their order) as further elements in between.

Example 2.10. In the following tree the bold-face formulae constitute a branch in the tree.



The particular shape of the formulae does not matter at the moment. You should just check that the sequence of formulae in boldface satisfies the definition of a branch: '$\neg\neg[Q \to R]$' is the end point of the branch. Taking this formula as the last formula and the topmost formula and all formulae in between yields a branch within the tree.

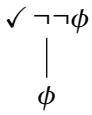
There are several rules that can be used to extend an existing tree; they allow one to write something at the end of one or several branches. A branch is closed if and only if the there is a bar at the bottom of the branch; otherwise the branch is unclosed (How branches can be closed will be explained below).

I pick one of the rules at random. Assume that a formula of the form '$\neg\neg\phi$' occurs in a tree and the formula is not yet ticked. Then there is a rule that allows one to write the formula ϕ at the end of any unclosed branch that contains this occurrence of '$\neg\neg\phi$' and to tick the occurrence of '$\neg\neg\phi$'.

You must write ϕ at the end of *any* unclosed branch that contains the occurrence of '$\neg\neg\phi$' in question. It is also mandatory that you tick the occurrence of '$\neg\neg\phi$'.

I have been talking about occurrences of formulae. The reason is that a formula may occur several times in a tree. The rule focuses on just one occurrence of the formula. If there is a second occurrence of '$\neg\neg\phi$' on another branch you do not have to add ϕ to this branch.

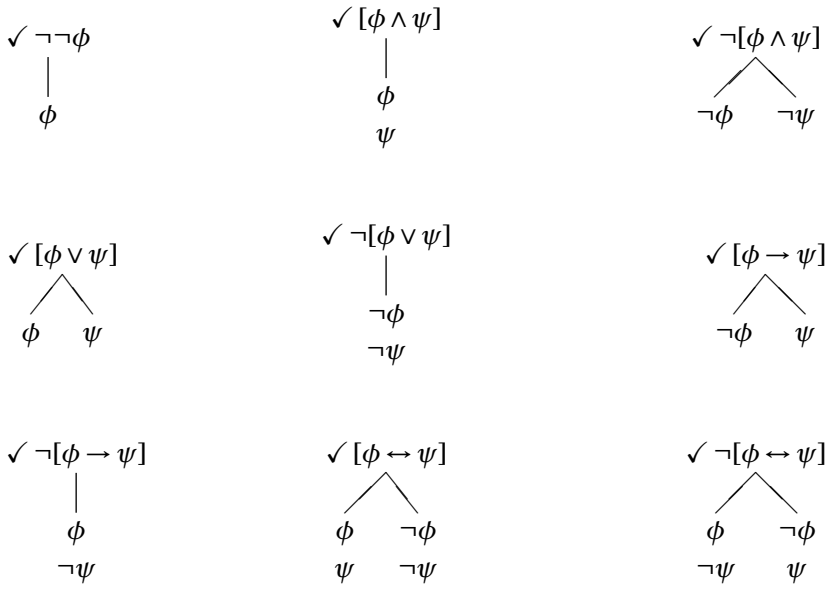
In the following I shall abbreviate this rule by writing:



Of course, there may be formulae on the branch between '$\neg\neg\phi$' and ϕ.

It is a common mistake to apply the rule not to the entire formula in a line, but only to parts of a formula. For instance, the formula does *not* allow you to pass from '$[P \wedge \neg\neg Q]$' to '$[P \wedge Q]$'. The rule cannot be applied to '$[P \wedge \neg\neg Q]$' at all, because this formula is of the form '$[\phi \wedge \psi]$' and not of the form '$\neg\neg\phi$'.

Other rules allow you to add two formulae at the end of branches. Still others allow you to add two formulae but on two different branches. Here is a list of all rules:



You may draw a line at the bottom of every branch on which a formula occurs together with its negation.

Definition 2.11. *A tree is a tableau if and only if all its branchings follow one of the above derivation rules.*

A branch is closed if and only if there is a line at the bottom of the branch; otherwise it is open. A tableau is closed if and only if all its branches are closed.

Definition 2.12. *Assume X is a finite set of formulae of $\mathscr{L}_1$. Then $X \vdash$ if and only if there is a closed tableau with all formulae in X at the top.*

Definition 2.13. *$X \vdash \phi$ if and only if $X, \neg\phi \vdash$.*

Thus we have $X \vdash \phi$ iff there is a closed tableau with all formulae in X and '$\neg\phi$' at the top. The elements of X are called *premisses*.

Instead of writing '$\{\psi_1, \ldots, \psi_n\} \vdash \phi$' one may also write '$\psi_1, \ldots, \psi_n \vdash \phi$'. That is, one may drop the set brackets around an enumeration of the premisses. Below '$X, \neg\phi$' stands for the set of all formulae in X plus the formula ϕ.

'$X \vdash \phi$' is read as 'X syntactically entails ϕ' or as 'ϕ is provable from the premisses in X', and '$\vdash \phi$' is read as 'ϕ is a (syntactic) theorem'.

X is allowed to be the empty set Ø and the same conventions apply. Thus $\vdash \phi$ if and only if there is a closed tableau with '$\neg\phi$' at the top. In this case ϕ is called *provable*.

'$X \vdash \phi$' is called a 'syntactic sequent'. H116

As an example I shall show

Example 2.14. $[P \leftrightarrow \neg[Q \to R]], [R \to [\neg Q \wedge S]] \vdash [[Q \vee P] \to \neg R]$

I shall now show how to generate a proof tree step by step.

In the first step one writes down all premisses, that is, all formulae in front of $\vdash$ and the negation of the conclusion:

$$[P \leftrightarrow \neg[Q \to R]]$$
$$[R \to [\neg Q \wedge S]]$$
$$\neg[[Q \vee P] \to \neg R]$$

Then one has the choice of applying a rule for one of the three formulae. I shall apply the rule for the formula at the bottom. The rule for formulae of the form '$\neg[\phi \to \psi]$' allows one to add ϕ and '$\neg\psi$'. One also has to tick the original formula:

$$
\begin{array}{c}
[P \leftrightarrow \neg[Q \rightarrow R]] \\
[R \rightarrow [\neg Q \wedge S]] \\
\checkmark\ \neg[[Q \vee P] \rightarrow \neg R] \\
| \\
[Q \vee P] \\
\neg\neg R
\end{array}
$$

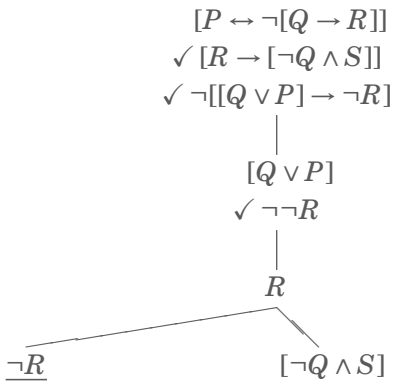
At this stage we could return to the first or second line, but I rather choose to continue with the last line and apply the rule for '$\neg\neg\phi$':

$$
\begin{array}{c}
[P \leftrightarrow \neg[Q \rightarrow R]] \\
[R \rightarrow [\neg Q \wedge S]] \\
\checkmark\ \neg[[Q \vee P] \rightarrow \neg R] \\
| \\
[Q \vee P] \\
\checkmark\ \neg\neg R \\
| \\
R
\end{array}
$$

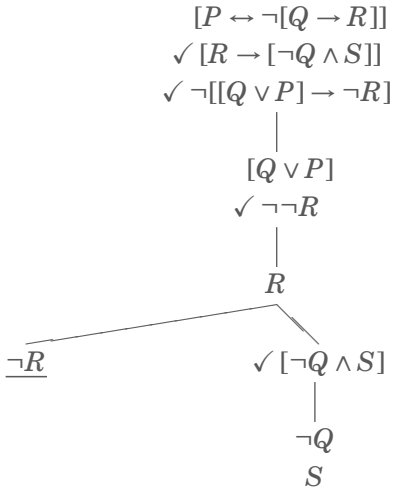
At this point I return to the second line (I could also have proceeded with '$[Q \vee P]$'):

$$
\begin{array}{ccc}
[P \leftrightarrow \neg[Q \rightarrow R]] & & \\
\checkmark\ [R \rightarrow [\neg Q \wedge S]] & & \\
\checkmark\ \neg[[Q \vee P] \rightarrow \neg R] & & \\
| & & \\
[Q \vee P] & & \\
\checkmark\ \neg\neg R & & \\
| & & \\
R & & \\
\swarrow & & \searrow \\
\neg R & & [\neg Q \wedge S]
\end{array}
$$

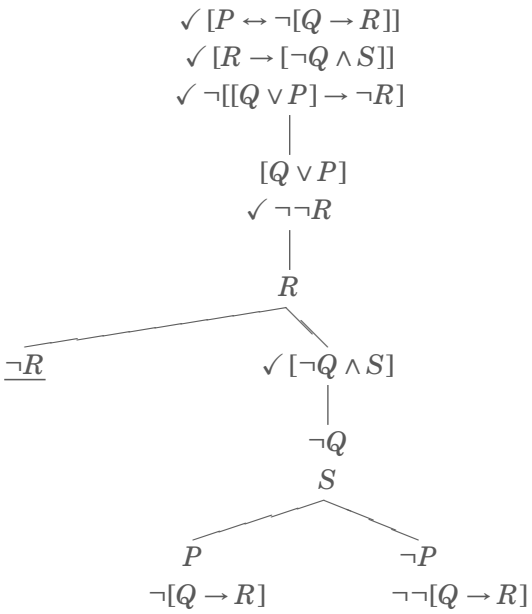
Now we may draw a line at the bottom of the left branch because it contains both R and '$\neg R$' (also the presence of '$\neg\neg R$' and '$\neg R$' would it make possible to close the branch):



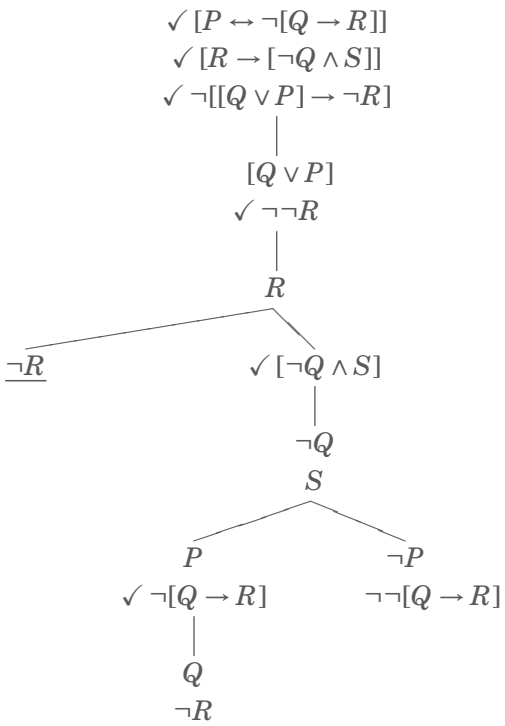
We continue with the right branch by applying the rule for '$[\phi \wedge \psi]$':



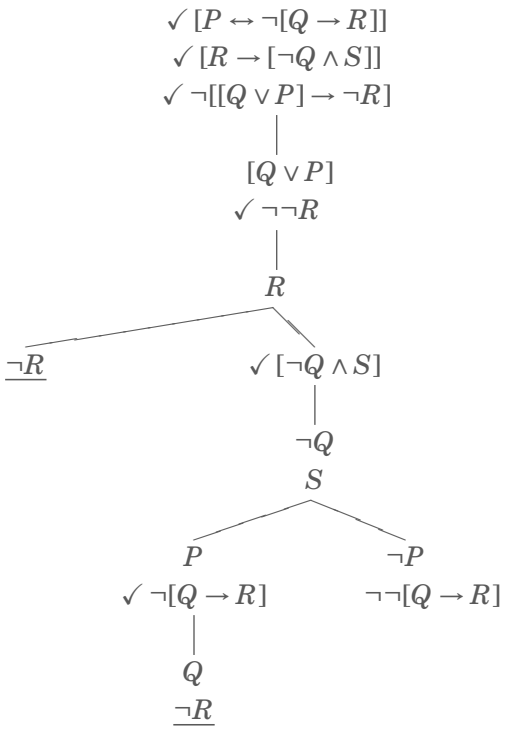
At this stage only two formulae are left to which we can apply a rule, namely the formula in the first line and '$[Q \vee P]$'. I turn to the first line:



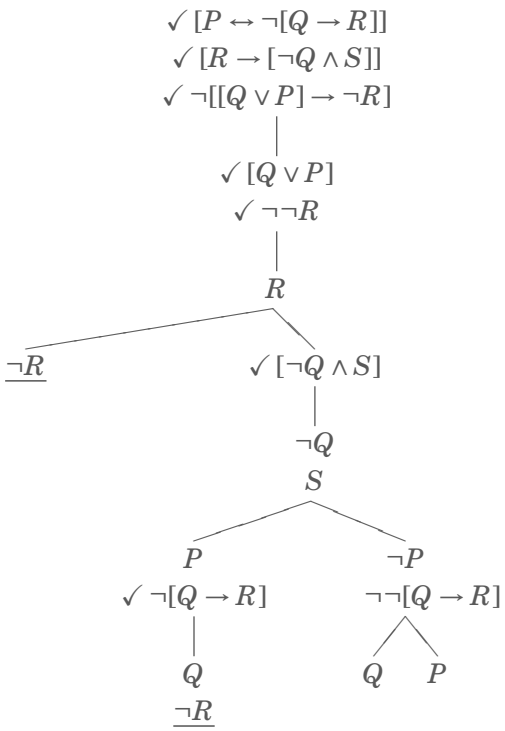
I continue with the branch in the middle and its last line:



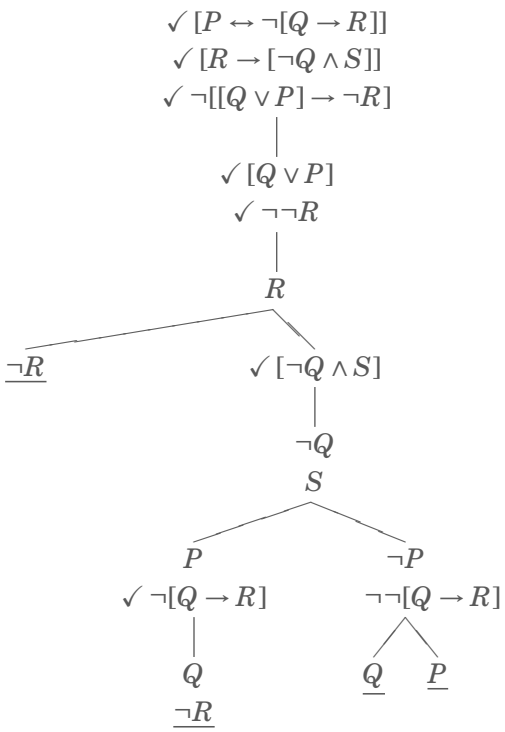
Now obviously the last branch can be closed because the last line is '$\neg R$' and R occurs on the branch.



Now there is only one open branch left. I use the fourth line in the tree:



Finally one may draw lines at the end of both open branches since both '$\neg Q$' and '$\neg P$' occur on that branch.



Since all branches are now closed, the claim in the example above is established.

I give now another example.

Example 2.15. $\vdash [\neg[P \leftrightarrow Q] \vee [P \leftrightarrow Q]]$

Since there is only a conclusion and no premisses, one has to start with the negation of the conclusion only. The rest should be obvious:

$$\begin{array}{c} \checkmark\ \neg[\neg[P \leftrightarrow Q] \vee [P \leftrightarrow Q]] \\ | \\ \neg\neg[P \leftrightarrow Q] \\ \underline{\neg[P \leftrightarrow Q]} \end{array}$$

The branch may be closed because both '$\neg\neg[P \leftrightarrow Q]$' and '$\neg[P \leftrightarrow Q]$' are on the same branch. There is no need to continue this proof and to decompose the formulae further.

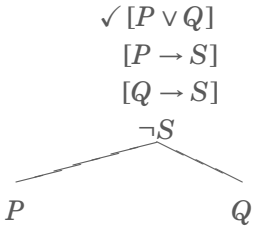
In general one saves some writing if one tries to avoid branchings as long as possible. For instance, if you can choose between applying the rule for '$\neg\neg\phi$' or the rule for '$[\phi \vee \psi]$', choosing the first will usually (but not always) produce a more economical proof.

Example 2.16. $[P \vee Q], [P \to S], [Q \to S] \vdash S$

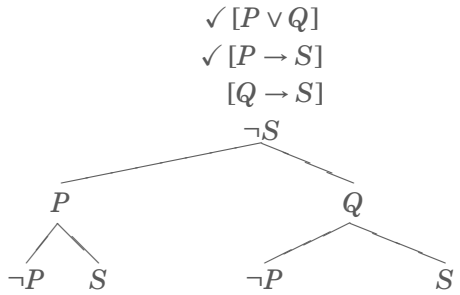
We start again in the obvious way:

$$
\begin{array}{c}
[P \vee Q] \\
[P \to S] \\
[Q \to S] \\
\neg S
\end{array}
$$

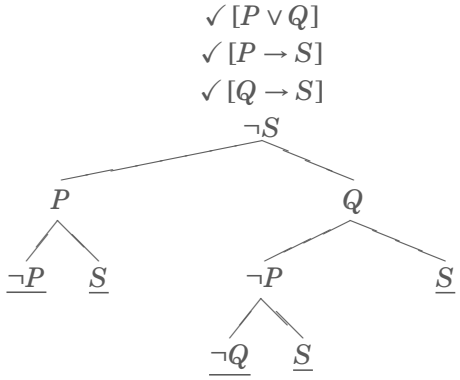
Now a branching cannot be avoided, so I turn to the first line:



Again the tree has to branch. The second line is used. It is mandatory that the branching is added to *both* open branches. If it were allowed to apply the rule to only one branch and to tick the original formula, then one migt end up with a tableau with all complex formulae ticked but still with open branches, although the formula is provable.



Obviously three branches can be closed and the third formula is then used to add a further branching to the only remaining open branch. In the next step the two open branches are closed.



Finally I mention a result without proof that relates $\models$ and $\vdash$:

H118

Theorem 2.17 (Adequacy)**.** *For all sets X of formulae of $\mathscr{L}_1$ and formulae ϕ the following holds:*

$$X \vDash \phi \textit{ iff } X \vdash \phi$$

The left-to-right direction is called the Completeness Theorem (for propositional logic); the right-to-left direction is called the Soundness Theorem (for propositional logic).

2.4 Formalisation

2.4.1 Truth-functionality

A sentence-functor is an expression containing English words and possibly Greek letters ϕ, ψ, χ and so on such that substituting the Greek letters by English declarative sentences yields an English declarative sentence.

'and', for instance, can be combined with 'It's raining.' and 'Oxford is small.' to 'It's raining and Oxford is small.' Some sentence-functor symbols take only one sentence, 'not' or 'it's probable that', for instance.

In $\mathscr{L}_1$ the connectives (truth-functor symbols) are sentence-functors. According to the semantics expounded in section 2.2, The truth or falsity of a sentence of a formula $[\phi \wedge \psi]$, for instance, in a structure **A** depends only on the truth and falsity of ϕ and ψ; the truth and falsity of ϕ and ψ in **A** uniquely determine the truth value of $[\phi \wedge \psi]$ in **A**. In $\mathscr{L}_1$ the truth and falsity of a formula in a structure always only on the truth values of the sentence letters in **A** occurring in the sentence. Exactly those sentence functors where the truth value of the compound sentence is uniquely determined by the truth values of the sentences replacing the Greek letters are truth-functional. The truth-functional sentence functors are also called truth functor.

In contrast, there are sentence functors in English that yield applied to sentences another sentence whose truth or falsity does not merely depend on the truth or falsity of its constituents.

'ϕ because ψ' is an example of a sentence functor where the truth value of the compound sentence is not fully determined by the truth values of the sentences substituted for ϕ and ψ. Whether 'ϕ because ψ' is true after a substitution does not only depend on whether ϕ and ψ are true. I shall give two examples. In both cases ϕ and ψ (or rather the sentences substituted for them) are true, but 'ϕ because ψ' is true in the first but false in the second case.

Imagine that I drop my laptop computer on the street,. The screen is broken and it is not functional anymore. So it is true, that my laptop computer does not work. The sentence

> My laptop computer does not work, because I dropped it.

is true: the laptop would still be functional if I had not dropped it. Moreover, it is true that the computer does not work and it is true that I dropped it. So we have an example where ϕ and ψ are true and 'ϕ because ψ' is also true.

It is also true that it is not plugged in, but the sentence

My laptop computer does not work, because it is not plugged in.

is false. For the computer would not work even if it were plugged in, because the battery is fully charged. It does not work, because it is broken, not because it is plugged in. Thus we have a case where ϕ and ψ are true and 'ϕ because ψ' is false. Therefore the sentence functor 'because' is not truth-functional.

Nevertheless we can determine a *partial* truth table for 'because'. We have already seen that in the first line of the following table, we do not have a unique truth value. But for the remaining lines the sentence 'ϕ because ψ' is always false. If ϕ is false, 'ϕ because ψ' can never be true, because 'ϕ because ψ' implies that ϕ is the case. Also if ψ is false, 'ϕ because ψ' must be false, because something that is false cannot be a reason for ϕ. So we arrive at the following partial truth table:

ϕ	ψ	ϕ because ψ
T	T	?
T	F	F
F	T	F
F	F	F

There might be a certain temptation to formalise 'because' by $\wedge$, but obviously this is not correct. Sometimes, however, one is more generous and formalises sentence functor that are not truth-functional by the connectives (or 'truth-functor symbols' of propositional logic. 'if—,then—', for instance, is usually formalised by $\rightarrow$, but it is usually not truth-functional.

Thus exactly those sentence functors with complete truth tables are truth-functional.

2.4.2 Logical form

In this section I show how to translate a given sentence of English into a formula of $\mathscr{L}_1$. Translation into a formal language is in general a highly nontrivial task. Here I shall give a practical procedure, which is far from being a precise account.

In the first step the English sentence is brought into a certain form. This is the difficult step.

The following procedure takes an English sentences and proceeds deeper and deeper into the structure of the sentence until propositional logic cannot be used anymore to provide a further analysis of the subsentences.

1. Check if the sentence can be reformulated as a sentence composed by a truth-functional sentence-functor in a natural way. If this is not possible the sentence is put in square brackets and not further analysed.

2. If the sentence can be reformulated as a sentence composed by a truth-functional sentence-functor in a natural way, rewrite the sentence in this way.

3. Given a sentence composed by a truth-functional sentence-functor, reformulate it with standard truth-functors. The standard truth functors are:

name	standard truth-functor	some other formulations
conjunction	and	but, although a komma between sentences
disjunction	or	
negation	it is not true that	not, none never etc.
arrow truth-functor	if—then—	given that—,—
biconditional	if and only if	exactly if

4. Surround the whole sentence by square brackets, unless it is a negated sentence, that is, a sentence starting with 'it is not true that'.

5. Apply the procedure starting at 1. again to the sentences that are surrounded by square brackets but do not contain further square brackets.

I illustrate the procedure with several examples:

Example 2.18. Rob and Tim will laugh, if the tutor can't pronounce Siobhan's name.

Leaving the paradoxes of material implications aside, the sentence is composed by the truth-functional sentence functor '—, if—'. There is no need for a reformulation according to 2. But '—, if—' is not a *standard* truth-functor. The standard truth-functor is 'if—then—'. I reformulate in accordance with 3. the sentence with this standard truth-functor:

> If the tutor can't pronounce Siobhan's name, then Rob and Tim will laugh.

In step 4. I surround the whole sentence by square brackets:

> [If the tutor can't pronounce Siobhan's name, then Rob and Tim will laugh]

Step 5. sends me back again to 1. There are 2 sentences left that are surrounded by square brackets and that do not contain square brackets, namely:

- the tutor can't pronounce Siobhan's name
- Rob and Tim will laugh

The first sentence contains a negation and I reformulate it with the standard truth-functor 'it is not true that—':

> it is not true that the tutor can pronounce Siobhan's name

Since the sentence starts with 'it is not true that', no brackets are inserted according to 4.

We still have to apply the procedure to the second sentence. 'Rob and Tim will laugh' is not a sentence composed by a truth-functional sentence-functor, but it can be reformulated as a sentence with a truth-functional sentence-functor as:

> Rob will laugh and Tim will laugh

This was step 2. 'and' is already in standard formulation, so I can skip step 3. By 4. I still need to put the direct subclauses in brackets:

> [Rob will laugh and Tim will laugh]

Thus the whole sentence reads now as follows:

> [If it is not true that the tutor can pronounce Siobhan's name, then
> [Rob will laugh and Tim will laugh]]

Now I have to start again with 1. The sentence 'the tutor can pronounce Siobhan's name' can be reformulated as a sentence composed by a truth-functional sentence-functor in a natural way. Thus it is put in square brackets according to 1.:

> [If it is not true that [the tutor can pronounce Siobhan's name],
> then [Rob will laugh and Tim will laugh]]

Also 'Rob will laugh' and 'Tim will laugh' cannot be reformulated as a sentence composed by a truth-functional sentence-functor. So they are also put into square brackets as required by 1.:

> [If it is not true that [the tutor can pronounce Siobhan's name],
> then [[Rob will laugh] and [Tim will laugh]]]

This is now the logical form of the sentence (in propositional logic).

Example 2.19. Unless the ignition is turned on and the fuels taps are opened, the engine will not start and I'll not be able to arrive in time.

For this example I shall give only the steps without commenting them. The numbers indicate which step of the procedure is applied. If there is nothing to do, e.g. because in 2. the sentence is already composed by a truth-functional sentence-functor, I shall skip it. In the first step I reformulate according to 3. 'unless' as 'if it is not true that—, then—'.

(3.) If it is not true that the ignition is turned on and the fuels taps are opened, then the engine will not start and I'll not be able to arrive in time.

(4.) [If it is not true that the ignition is turned on and the fuels taps are opened, then the engine will not start and I'll not be able to arrive in time]

(4.) [If it is not true that the ignition is turned on and the fuels taps are opened then [the engine will not start and I'll not be able to arrive in time]]

(3.) [If it is not true that the ignition is turned on and the fuels taps are opened then [it is not true that the engine will start and it is not true that I'll be able to arrive in time]]

(4.) [If it is not true that [the ignition is turned on and the fuels taps are opened] then [it is not true that the engine will start and it is not true that I'll be able to arrive in time]]

(1.) [If it is not true that [the ignition is turned on] and [the fuels taps are opened] then [it is not true that [the engine will start] and it is not true that [I'll be able to arrive in time]]]

It is not necessary to treat all all subclauses simultaneously, as I have done it here. You could first analyse the first part in full detail and then turn to the second. The result ought to be the same.

2.4.3 From logical form to formal language

Once the logical form of a sentence has been determined, the translation into the language $\mathscr{L}_1$ of propositional logic is purely mechanical.

In order to translate the logical form of an English sentence into $\mathscr{L}_1$ apply the following procedure:

1. Replace standard truth functors by their respective symbols according to the following list:

standard truth-functor	symbol
and	$\wedge$
or	$\vee$
it is not true that	$\neg$
if—then—	$\rightarrow$
if and only if	$\leftrightarrow$

2. Replace every (English) sentence including its brackets by a sentence letter. Use different sentence letters for distinct sentences and the same sentence letter for multiple occurrences of the same sentence.

3. give a list of all sentence letters in the resulting formula with the respective sentences they have replaced.

I shall carry out this procedure on Example 2.18. Its logical form has been determined as:

> [If it is not true that [the tutor can pronounce Siobhan's name],
> then [[Rob will laugh] and [Tim will laugh]]]

First I replace all standard truth functors by the respective symbols, as required by 1.:

> [$\neg$ [the tutor can pronounce Siobhan's name] $\rightarrow$ [[Rob will laugh]
> $\wedge$ [Tim will laugh]]]

In step 2. the sentences are replace by sentence letters:

$$[\neg P \rightarrow [Q \wedge R]]$$

Here is the list of step 3.:

P: the tutor can pronounce Siobhan's name

Q: Rob will laugh

R: Tim will laugh

3 Predicate logic

3.1 The syntax of the language of predicate logic

Hodges (2001) does not specify the syntax of predicate logic. There are some snags in Hodges' approach. For instance, one would expect that P in Pxy and in Px are the same predicate letter. However, they are not.

In general Hodges does not carefully distinguish between the formal and the natural language.

Definition 3.1 (predicate letters)**.** *All upper case Latin letters 'A^0', 'B^0', 'C^0', 'A^0', 'B_0^0', 'C_0^0',...,'A_8^{24}',... with an arbitrary natural number or no number as subscript and with some number as superscript are predicate letters.*

The value of the upper index of a predicate letter is called its arity. The predicate letter 'P_4^3', for example, has arity 3. A predicate letter of arity 3, for example, is sometimes called a 3-place predicate letter.

Definition 3.2 (variables)**.** *'x', 'y', 'z', 'x_1', 'y_1', 'z_1', 'x_2',... are variables.*

Definition 3.3 (individual constants)**.** *'a', 'b', 'c', 'd', 'a_1', 'b_1', 'c_4', 'd_1', 'a_2',... are individual constants.*

Hodges calls the individual constants 'designators' and 'proper names' if I understand him correctly. This is very unfortunate, because he also calls certain expressions of the natural language 'designators'.

Definition 3.4 (atomic formulae)**.** *If P is a predicate letter of arity n and $t_1,\ldots,t_n$ are variables or individual constants, then '$Pt_1\ldots t_n$' is an atomic formula. If t_1 and t_2 are variables or individual constants, then '$t_1 = t_2$' is also an atomic formula.*

'$G_5^3xd_4y$', '$x = a$' and 'G^2xx', for instance, are atomic formulae.

In an atomic formula the arity of a predicate letter is obvious: it is the numbers of variables and individual constants following the predicate letter. Therefore we adopt the following convention:

> Superscripts of predicate letters may be dropped in atomic formulae.

Definition 3.5. *A quantifier is an expression '$\forall v$' or '$\exists v$' where v is a variable.*

There are alternative symbols for '$\forall$' and '$\exists$' (which will not be used here or in examinations). '$\bigwedge v$', 'Πv' and '(v)' are sometimes used instead of '$\forall v$', and '$\bigvee v$' and 'Σv' instead of '$\exists v$'.

$\mathscr{L}_1$ was used for the language of propositional logic; the language of predicate logic is labelled $\mathscr{L}_2$.

Definition 3.6 (formula of $\mathscr{L}_2$)**.**

(i) All atomic formulae are formulae of $\mathscr{L}_2$.

(ii) If ϕ and ψ are formulae of $\mathscr{L}_2$, then '$\neg\phi$', '$[\phi \wedge \psi]$', '$[\phi \vee \psi]$', '$[\phi \to \psi]$' and '$[\phi \leftrightarrow \psi]$' are formulae of $\mathscr{L}_2$.

(iii) If v is a variable and ϕ is a formula with an occurrence of v but without an occurrence of a quantifier '$\forall v$' or '$\exists v$', then '$\forall v\phi$' and '$\exists v\phi$' are formulae of $\mathscr{L}_2$.

Examples of formulae of the language $\mathscr{L}_2$ of predicate logic are:

- $\forall x[P^2xa \to Q^1x]$
- $\neg[\forall x \forall y[P^3_2axx \wedge \exists z P^3_2zyc] \wedge P^0]$
- P^3xya
- $[[\forall z \exists y Rzy \leftrightarrow \exists z Rzz] \wedge \forall z Pz]$ This is a formula in accordance with Definition 3.6 (iii), although there are three occurrences of quantifiers involving z.

Do not try to understand these expressions, just check that these expressions are formulae of $\mathscr{L}_2$ according to the definition above.

The following two displayed formulae are *not* formulae of $\mathscr{L}_2$:

$$\forall x[Py \to Qy]$$

This is not a formula, because the variable x has no occurrence in '$[Py \to Qy]$'.

$$\exists x[Pxa \wedge \forall x[\neg Gx \vee Qx]]$$

This is not a formula, because the quantifier '$\forall x$' occurs in '$[Pxa \wedge \forall x[\neg Gx \vee Qx]]$', which is excluded in Definition 3.6 (iii).

Many other authors use a slightly different definition of a formula according to which the two expressions above would be formulae.

Again superscripts of predicate letters may be skipped. So for the first formula one would usually write '$\forall x[Pxa \to Qx]$'.

H176ff

Definition 3.7 (scope of a quantifier)**.** *The scope of a quantifier within a given formula ϕ is the smallest formula within ϕ that contains this quantifier.*[1]

Definition 3.8 (bound occurrence of a variable)**.** *An occurrence of a variable v is bound if and only if it is in the scope of a quantifier '$\forall v$' or '$\exists v$'.*

An occurrence of a variable is free if and only if it is not bound.

Definition 3.9 (closed formulae)**.** *A formula is closed if and only if all occurrences of variables are bound in the formula.*

It is convenient to stipulate that all occurrences of individual constants in formulae are also free.

I list some examples. The underlined occurrences of variables are free in the respective formulae.

- $[\forall x[Px\underline{y} \rightarrow Qax] \leftrightarrow P\underline{x}\underline{y}]$
- $\exists z \neg [Pa\underline{x} \wedge \neg [\exists x Px \vee Qz\underline{x}]]$
- $[[P\underline{x} \wedge A\underline{x}\underline{y}] \wedge R\underline{x}\underline{y}]$

The occurrences of variables that immediately follow the quantifier symbols '$\forall$' and '$\exists$' are usually not considered. One can take all of them as bound occurrences or as neither bound nor free.

3.2 Formalisation

3.2.1 Atomic sentences

Simple sentences like 'Michael Schumacher is fast.' cannot be further analysed in propositional logic; they are formalised as sentence letters.

In predicate logic these sentence can be further analysed. Proper names like 'Michael Schumacher' denoting a fixed singular object (or at least intended to denote a single object) are formalised as individual constants. We shall look at the exceptions below and concentrate for the moment on the straightforward cases.

So 'Michael Schumacher' may be formalised as the individual constant 'a'. The expression 'is fast' is a predicate and is formalised by a predicate letter like 'P^1_3'. The letter itself and the subscript '3' are not important here, but the

[1] To be precise, I should talk about occurrences of quantifiers and formulae. In the formula '$[\exists x Px \wedge \exists x Qx]$' the quantifier '$\exists x$' occurs twice. The scopes of the two occurrences are obviously different. However, I shall suppress the reference to particular occurrences if it is clear which occurrence is discussed.

superscript '1' *is* important, because 'is fast' takes one singular term, so it is a unary predicate, that requires a unary predicate letter for its formalisation.

The syntax of $\mathscr{L}_2$ requires that predicate letters precede the accompanying terms. Thus

Schumacher is fast.

is formalised as the following sentence of the language $\mathscr{L}_2$ of predicate logic:

$$P^1_3 a$$

The formalisation must also specify the translation of all involved predicate letters and individual constants:

$P^1_3 x$: x is fast

a: Michael Schumacher

Of course, following the conventions superscripts may be skipped. So one may also write '$P_3 a$'.

Another example is 'Michael Schumacher overtakes Alonso'. This sentence involves two singular term, 'Michael Schumacher' and 'Alonso'. In this case we employ a binary predicate letter and formalise the sentence as the following sentence of $\mathscr{L}_2$:

$$Q^2 a b$$

Of course the translation is the following:

$Q^2 xy$: x overtakes y

a : Michael Schumacher

b : Alonso

In $\mathscr{L}_2$ predicate letters have a fixed arity, while it varies in English. In the following sentence 'stabbed' woul be formalised with a 4-place predicate letter:

In 44 B.C. Brutus stabbed Caesar in Rome.

If 'in Rome' were deleted a ternary predicate letter would suffice. We shall return to the problem of variable arity later.

An example of a complex sentence is the following:

Schumacher is fast and overtakes Alonso.

In a first step towards a formalisation we can reexpress this as:

Schumacher is fast and Schumacher overtakes Alonso.

This gives the formalisation:

$$[P_3^1 a \wedge Q^2 ab]$$

The translations are as above.

Obviously the same method applies also to certain sentences containing personal pronouns like

Schumacher is fast and he overtakes Alonso.

Here one can replace the personal pronoun 'he' by the proper name 'Schumacher' and then proceed as above with the formalisation. Such (uses of) pronouns are called 'lazy'. Using 'he' only saves us the effort of repeating the name 'Schumacher'.

However, other occurrences of personal pronouns are not as easily eliminated, as will be shown in the next section.

There is a special predicate that is always translated by the same symbol namely identity. The sentence

New College is the College St. Mary of Wincester.

would be translate into

$$a = b$$

a : New College

b : the College St. Mary of Wincester

There is no need to specify the translation of '=', because it is always translated as identity. The identity symbol is also used in other formalisation:

New College is different from Oriel College.

is simply the negation of

New College is Oriel College.

Later identity will be used for formalising that seem to have nothing to do with relation.

3.2.2 Quantification

Quantified sentences like 'All men are mortal' were scrutinised by logicians since antiquity. It took over 2000 years to develop the modern theory of quantification in predicate logic. In this section I shall try to motivate the

modern treatment of quantified sentences and show how quantified sentences are formalised.

In English quantification can be expressed in various ways. The following two sentences are equivalent, although the methods of expressing quantification are different:

(i) All four-legged vertebrates with gills are amphibians.

(ii) If a vertebrate has four legs and if it has gills, it is an amphibian.

Logicians are seeking a *uniform* way to express quantification. For a very long time they took (i) as the pattern after which they modelled the formalisation of all quantified sentences. In modern predicate logic (ii) is closer to its formalisation.

Above I have claimed that not all occurrences of personal pronouns can be eliminated easily: the 'it' in (ii) is an example. Obviously, (ii) cannot be rephrased as 'If a vertebrate has four legs and if it has gills, a vertebrate is an amphibian.', rather we would have to say 'If a vertebrate has four legs and if it has gills, then this very same vertebrate is an amphibian.'. But 'this' is a pronoun again that cannot be eliminated by a proper name. So the 'it' in the example is not a lazy pronoun.

In the following example several quantifications are involved and personal pronouns are employed to express quantification.

> If a student borrows a book from the library, he has to show his university card to Ms Smith. If she approves he may leave the library with it unless it is expired.

It is hard to rephrase this without using pronouns. The example shows that pronouns can refer back to phrases that occurred in preceding sentences. In the example both occurrences of 'he' refer back to 'a student'. Method (i) for expressing quantification is far less flexible. This is the main reason why modern predicate logic models quantification along the lines of (ii) and not (i).

The example shows also limits of method (ii): There are two occurrences of the pronoun 'it', and the first occurrence probably refers back to 'a book', the second probably refers back to 'university card'. This is not clear from the syntax of the example; one needs to look into the content of the sentence to determine the references of the two occurrences of 'it'.

There are other pronouns whose reference is clear from the syntax of the example alone. The first occurrence of 'he' *has* to refer to the student, because the other phrases in question, 'a book' and 'he library', would require 'it' rather than 'he' as pronouns. Thus in English the cross-referencing can be disambiguated by the gender and number (singular or plural) of the pronoun. But the example of 'it' shows that this method does not remove all ambiguities, because we cannot use always 'new' pronouns.

The ambiguities of cross-referencing can be removed by attaching indices to the pronouns. Since 'she' is a lazy pronoun, it can be be replaced by 'Ms Smith'.

> If a student$_1$ borrows a book$_2$ from the library, he$_1$ has to show his$_1$ university card$_3$ to Ms Smith. If Ms Smith approves he$_1$ may leave the library with it$_2$ unless it$_3$ is expired.

By convention the pronoun with index n refers back to the phrase where the index n appeared first. In the presence of indices, there is no longer a need for indicating the gender of the pronoun. Therefore the pronouns can be replaced by so called variables without any loss of information:

> If a student$_1$ borrows a book$_2$ from the library, x_1 has to show x_1's university card$_3$ to Ms Smith. If Ms Smith approves x_1 may leave the library with x_2 unless x_3 is expired.

The language $\mathscr{L}_2$ of predicate logic contains individual variables (see section3.1). Thus by combining the techniques from section 2.4.3 and section 3.2.1, one can formalise the example—except for the phrases 'a student$_1$', 'a book$_2$', 'x_1's' and 'university card$_3$'. They cannot be formalised by an individual constant, because, for instance, 'a student$_1$' does not refer to a particular person or object like 'Michael Schumacher' in section 3.2.1. We need some way to express in the formal language that the variable x_1 ranges over students, x_2 over books in the library, x_3 over university cards.

In the above example 'he' refers back to 'a student'; therefore the pronoun 'he' ranges over students. Using such phrases like 'a student' and pronouns referring back to 'a student', one can make claims about *all* students.

Because of its complexity I leave the example and return to the initial examples i and ii:

(i) All four-legged vertebrates with gills are amphibians.

(ii) If a vertebrate has four legs and if it has gills, it is an amphibian.

I have argued that expressing quantification with phrases like 'a student'—or here 'a vertebrate'—plus personal pronouns is more flexible and more amenable to complex quantifications. If we were going to formalise this in a formal language, we would need expressions corresponding to 'a vertebrate', 'a student' and so on. This can be done, but there is a further simplification. In order to express

> Every tortoise is a reptile.

we do not need 'a tortoise'. We could use 'a vertebrate' again and say:

> If a vertebrate is a tortoise, it is a reptile.

So if we already have a more inclusive quantificational phrase like 'a vertebrate', we do not need the more restrictive phrase 'a tortoise', because the restriction can the expressed with a suitable 'if'-sentence.

Therefore in order to restrict the number of quantificational phrases, we should use more inclusive phrases. 'something' is *very* inclusive. Using something we can reexpress (ii) as:

> If something (whether it is a person, material object or something else) is a four-legged vertebrate and if it has gills, it is an amphibian.

We do not need any quantificational phrase beyond 'something' because it is most inclusive. Of course, in order to avoid ambiguitities in some more complicated examples, 'something' must be indexed, although the present example does not give rise to possible confusions. Here we would simply have the following:

> If something$_1$ is a four-legged vertebrate and if it$_1$ has gills, it$_1$ is an amphibian.

We are now already fairly close to the way quantification is expressed in the language $\mathscr{L}_2$ of predicate logic. But there is still one problem left, which is illustrated by the following sentence:

(3.1) Something is close to everything.

The sentence is ambiguous: Does it mean that there is one single object that is close to every object, or does it mean that everything is close to something, though not necessarily to the same object. Thus there are the following two readings of the sentence:

1. There is something$_1$ such that x_1 is close to everything.
2. Everything$_1$ is close to something

Once one has decided whether to choose (i) or (ii), one can go further and analyse the remaining parts of sentences in the following way, respectively:

1. There is something$_1$ such that for everything$_2$ it$_1$ is close to it$_2$.
2. For everything$_1$ there is something$_2$ such that it$_1$ is close to it$_2$

In $\mathscr{L}_2$ the same strategy is used: one utilises expressions corresponding to 'there is something$_1$ such that' and 'for everything$_1$:'. These devices are called the existential and the universal quantifiers and expressed in $\mathscr{L}_2$ as $\exists x_1$ and

$\forall x_1$, respectively.[2] Thus the ambiguity in (3.1) arises because the order of the quantifiers is unclear, that is, whether the existential quantifier precedes the universal quantifier or vice versa.

Now we can formalise 1. 'There is something$_1$ such that' becomes $\exists x_1$, while 'for everything$_2$' becomes $\forall x_2$. 'it$_1$' becomes x_1 and 'it$_2$' becomes x_2, so we have:

$\exists x_1 \forall x_2$ x_1 is close to x_2.

Using the following translation as in section 3.2.1

$$Pxy: \ x \text{ is close to } y$$

we obtain the following $\mathscr{L}_2$-sentence:

$$\exists x_1 \forall x_2 P x_1 x_2$$

It would be equally correct to write:

$$\exists x \forall y P x y \qquad \text{or } \exists z \forall x P z x$$

One has only to be careful to make sure that the respective quantifier refers to the correct variable.

We can also formalise 2. in the following way:

$$\forall x \exists y P x y$$

Obviously merely the order of the quantifiers has been reverted. This way we can distinguish between 1. and 2.

Here is another example:

Every dog is mortal.

This is a claim about all dogs. Thus one starts with a universal quantifier and expresses the restriction to dogs by an 'if'-clause:

For everything$_1$: if x_1 is a dog, x_1 is mortal.

First I specify the translation of the predicate letters:

$$Dx: \ x \text{ is a dog}$$
$$Mx: \ x \text{ is mortal}$$

'x_1 is a dog' thus becomes 'Dx_1' and 'x_1 is mortal' becomes 'Mx_1'. Using the techniques of section 2.4.2, 'if x_1 is a dog, x_1 is mortal' becomes '$[Dx_1 \rightarrow$

[2] The upside down 'A' reminds of 'all', while $\exists$ reminds of 'there **e**xists'.

Mx_1]'. Formalising 'For everything$_1$' as the universal quantifier '$\forall x_1$' yields the following translation of the whole sentence:

$$\forall x_1[Dx_1 \rightarrow Mx_1]$$

Thus the conditional is used to restrict the quantification to certain objects, here dogs. One says: 'for everything: *if* it is a dog it is mortal.' Conjunction serves the same purpose in the case of existential quantification. The sentence

There is a mortal dog.

would be formalised by $\exists x_1[Dx_1 \wedge Mx_1]$, that is, 'there is something that is a dog *and* that is mortal.' The formula $\exists x_1[Dx_1 \rightarrow Mx_1]$, in contrast, would say: 'There is something that is mortal, if it is a dog.' This is clearly something else. The sentence would already be true if there were a single stone and no dogs at all, because the sentence 'if the stone is a dog, it would be mortal' is true; thus there would exist something (namely the stone) that is mortal, if it is a dog. Thus one cannot use the conditional for restricting the quantification in the case of existentially quantified sentences. Similarly one cannot use conjunction for restricting quantification in the case of universal quantification.

3.2.3 Identity and definite descriptions

'is' can play various rôles. It can be used to express predication such as in 'Snow *is* white' or 'Jane *is* a classicist'. In these cases 'is' forms part of the predicate. 'is a classicist' will be formalised as a single predicate letter.

In other cases 'is' is used to express identity such as in 'Ratzinger is Benedict XVI.' or 'St. Mary's of Winchester college is New College'. Sentences of this kind are formalised as $a = b$ or other individual constants.

Identity is not only useful for formalising overt identity statements as in the above examples. Using identity we can also express that there is a certain number of objects. Assume the following interpretation:

Cx: x is a college

Then the claim that there is at least one college can be expressed by existential quantification as $\exists x Px$. If we want to say that there are at least two colleges, it does not suffice to say $\exists x \exists y[Cx \wedge Cy]$, because this says merely that something is a college and something is a college; it does *not* say that something is a college and something *else* is a college. But the latter can be expressed with identity:[3]

$$\exists x \exists y [Cx \wedge Cy \wedge \neg x = y]$$

[3] I am using the bracket conventions in he following, because these brackets drive me crazy.

So this formula actually expressed that there are at least two colleges. Of course the trick works also with three:

$$\exists x \exists y \exists z \, [Cx \wedge Cy \wedge Cz \wedge \neg x = y \wedge \neg x = z \wedge \neg y = z]$$

Obviously the formulae become very long, because one must express that all the objects are mutually different.

By using identity we can also express that there is *at most* one college by saying that if x and y are colleges, then x and y are identical:

$$\forall x \forall y [[Cx \wedge Cy] \rightarrow x = y]$$

Again this works also for 'at most two', 'at most three' and so on. 'There are at most two colleges' can be formalised as

$$\forall x \forall y \forall z [[Cx \wedge Cy \wedge Cz] \rightarrow [x = y \vee y = z \vee x = z]]$$

Combining both, 'at least' and 'at most', using a conjunction we can say that there are exactly three colleges. 'There is exactly one college' becomes

$$\exists x \, Cx \wedge \forall x \forall y [[Cx \wedge Cy] \rightarrow x = y]$$

This can be also expressed by the following logically equivalent formula:

$$\exists x \forall y [Cy \leftrightarrow x = y]$$

With these tricks at our disposal we can tackle a problem that arises from the formalisation of so called definite descriptions.[4] 'the king of France', 'the tallest tutor of New College' are examples of definite descriptions. Definite description cannot be adequately formalised by an individual constant. For instance, if 'the king of France' were the interpretation of the constant a and 'is bald' were the interpretation of P, then the formula:

$$\neg Pa$$

appears to the formalisation of the English sentence

> It is not the case that the king of France is bald.

Below we shall see that $\neg Pa \vdash \exists x \neg Px$ is a correct sequent. But in English the sentence 'Something is not bald' does *not* follow from 'It is not the case that the king of France is bald.' The latter sentence does *not* say that there is actually a king of France; it only says that he bald if there is a king of France.

In general individual constant should only be used for designator that are 'guaranteed' to refer to a single object. There is no such restriction on definite descriptions. We could now introduce a new symbol for definite description; the Greek letter iota ι is commonly used for that purpose. Assume the following interpretation:

[4] This analysis of definite descriptions is due to Bertrand Russel in *On denoting*.

Kxy: x is the king of y

f: France[5]

'the king of France' would then be formalised as $\iota x Kxf$.

It is not the case that the king of France is bald.

This sentence would then be formalised by

$$\neg P\, \iota x Kxf$$

Russell observed that we do not really need a new device for expressing this sentence: we do not need the expression $\iota x Kxf$. Russell took the sentence

The king of France is bald.

to say:

(i) There is exactly one king of France.

(ii) All kings of France are bald.

From the discussion above we know how to express (i) and (ii) does not cause any problems at all. So the formalisation of 'The king of France is bald' is the following formula:

$$\exists x \forall y[Kyf \leftrightarrow x = y] \wedge \forall x[Kxf \rightarrow Px]$$

Note that Russell does *not* say that the definite description $\iota x Kxf$ corresponds to a certain expression of the language of predicate logic: there is no such expression. Rather Russell shows how to eliminate $\iota x Kxf$ from a whole sentence. This makes Russell definition of the definite description operator ι a so called context definition.

The whole sentence

It is not the case that the king of France is bald.

is then formalised as

$$\neg[\exists x \forall y[Kyf \leftrightarrow x = y] \wedge \forall x[Kxf \rightarrow Px]]$$

Now we return to the discussion of 'is' at the beginning. In the following sentence 'is' expresses identity:

Louis is the king of France.

[5] France probably deserves an individual constant.

This may be formalised as follows:

$$b = \iota x K x f$$

Here b stands for 'Louis', of course. This formula can then by paraphrased by Russell's technique as the following sentence without a ι-operator:

$$\exists x \forall y [K y f \leftrightarrow x = y] \wedge \forall x [K x f \rightarrow x = b]$$

This can also be written much shorter in the following way:[6]

$$\forall x [x = b \leftrightarrow K x f]$$

Especially identity statements involving definite descriptions may be confusing. Consider the following two sentences:

(i) Jane is a classicist.

(ii) Jane is the classicist.

In the first sentence 'is' does not express identity. As remarked above, it expresses predication. (ii), however, is an identity statement: 'Jane' is a proper name, while 'the classicist' is a definite description. If Qx is interpreted as 'x is a classicist' and c by 'Jane', sentence (i) becomes Qc, while (ii) becomes the following formula:

$$\forall x [x = c \leftrightarrow Q x]$$

This says that there is exactly one classicist, and he or she is identical with Jane.

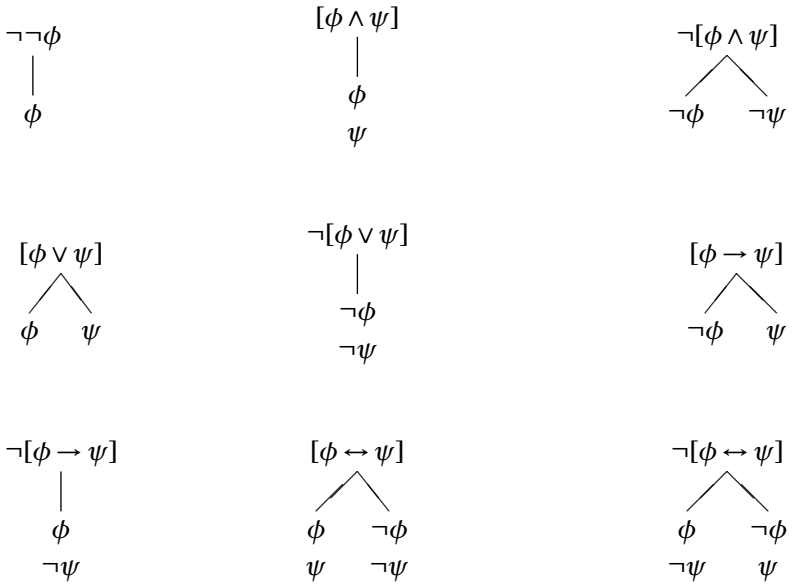
3.3 Proofs

> Von einer logisch vollkommenen Sprache [. . .] is zu verlangen, daß jeder Ausdruck, der [. . .] als Eigenname gebildet ist, auch in der Tat einen Gegenstand bezeichne [. . .].
>
> Gottlob Frege, *Über Sinn und Bedeutung*

Most definitions of section 2.3 carry over from propositional logic. Formulae of $\mathscr{L}_1$ are replaced by closed formulae of the language $\mathscr{L}_2$ of predicate logic. Moreover, the formulae are not ticked and it is not necessary to extend every branch containing the occurrence of the formula in question. H190ff

The rules for quantifiers look as follows:

[6] This method for simplifying sentences involving definite descriptions works only if an identity is claimed that has an individual constant on one side and a definite description on the other.



There are additional rules for identity:

$$\begin{array}{c}\phi\\ D = E\\ \vert\\ \psi\end{array}$$

provided that the individual constant D occurs in ϕ, and ψ is the result of replacing one or more occurrences of D in ϕ by occurrences of E.

Here it is only required that ϕ and '$D = E$' occur on the branch; it is not required that '$D = E$' has ϕ directly above it. It is also allowed that '$D = E$' occurs before ϕ on the branch.

The following rule differs from the first only in the order of the individual constants D and E.

$$\begin{array}{c}\phi\\ E = D\\ \vert\\ \psi\end{array}$$

provided that the individual constant D occurs in ϕ, and ψ is the result of replacing one or more occurrences of D in ϕ by occurrences of E.

The remaining rules concern the quantifiers:

$$\begin{array}{c}\forall v\phi\\ \vert\\ \psi\end{array}$$

provided that there is an individual constant D which has already occurred in the branch above ψ, and ψ is the result of replacing every free occurrence of the variable v in ϕ by D.

$$\begin{array}{c}\exists v\phi\\ \vert\\ \psi\end{array}$$

provided that ψ is the result of replacing every free occurrence of v in ϕ by the individual constant D and D has not occurred anywhere in the branch above ψ.

$$\begin{array}{c} \neg\forall v\phi \\ | \\ \exists v\neg\phi \end{array} \qquad\qquad \begin{array}{c} \neg\exists v\phi \\ | \\ \forall v\neg\phi \end{array}$$

Hodges (2001) mentions a rule VII, but he does not allow it to be used. This rule must *not* be used according to the examination regulations, unless you are explicitly allowed to do so.

The rule for drawing bars at the end of branches is as follows:

> You may draw a line at the bottom of every branch on which a formula occurs together with its negation or on which '$\neg D = D$' occurs for some individual constant D.

The definition of a tableau, a closed tree and so on are as in section section 2.3.

Definition 3.10. *Assume X is a finite set of formulae of $\mathscr{L}_2$ and ϕ is a formula of $\mathscr{L}_2$.*

(i) $X \vdash$ *if and only if there is a closed tableau with all formulae in X at the top.*

(ii) $X \vdash \phi$ *if and only if* $X, \neg\phi \vdash$.

Thus $X \vdash \phi$ if and only if there is a closed branch with all formulae in X and '$\neg\phi$' at the top. The elements of X are called *premisses*. '$X \vdash \phi$' is read as 'ϕ is provable from the premisses in X'.

As above I shall give a sample proof.

Example 3.11. $\vdash \forall x\forall y[x = y \rightarrow y = x]$

We start as always with the negation of the conclusion (usually we would also write down all the premisses, but in the present example there are not any premisses):

$$\neg\forall x\forall y[x = y \rightarrow y = x]$$

There is only one rule that applies to negated existential formulae. So we arrive at the following:

$$\begin{array}{c} \neg\forall x\forall y[x = y \rightarrow y = x] \\ | \\ \exists x\neg\forall y[x = y \rightarrow y = x] \end{array}$$

Now the existential quantifier '$\exists x$' is dropped and the variable 'x' is replaced by the individual constant a. This is only allowed because the individual constant a has not occurred so far in the branch so far.

$$\neg\forall x\forall y[x = y \to y = x]$$
$$\vert$$
$$\exists x\neg\forall y[x = y \to y = x]$$
$$\vert$$
$$\neg\forall y[a = y \to y = a]$$

The next quantifier is treated similarly:

$$\neg\forall x\forall y[x = y \to y = x]$$
$$\vert$$
$$\exists x\neg\forall y[x = y \to y = x]$$
$$\vert$$
$$\neg\forall y[a = y \to y = a]$$
$$\vert$$
$$\exists y\neg[a = y \to y = a]$$
$$\vert$$
$$\neg[a = b \to b = a]$$

Here a *new* individual constant must be used. It would not have been admissible to add '$\neg[a = a \to a = a]$'. In the next step we apply the rule for '$\neg[\phi \to \psi]$':

$$\neg\forall x\forall y[x = y \to y = x]$$
$$\vert$$
$$\exists x\neg\forall y[x = y \to y = x]$$
$$\vert$$
$$\neg\forall y[a = y \to y = a]$$
$$\vert$$
$$\exists y\neg[a = y \to y = a]$$
$$\vert$$
$$\neg[a = b \to b = a]$$
$$\vert$$
$$a = b$$
$$\vert$$
$$\neg b = a$$

Next we apply the first rule for identity. It allows to replace the occurrence of b in '$\neg b = a$' by an occurrence of a.

$$\vert$$
$$\neg\forall x\forall y[x = y \to y = x]$$

$$
\begin{array}{c}
\vdots \\
\exists x \neg \forall y [x = y \to y = x] \\
\vdots \\
\neg \forall y [a = y \to y = a] \\
\vdots \\
\exists y \neg [a = y \to y = a] \\
\vdots \\
\neg [a = b \to b = a] \\
\vdots \\
a = b \\
\vdots \\
\neg b = a \\
\vdots \\
\underline{\neg a = a}
\end{array}
$$

The last rule allows to draw a line at the end of the branch. Thus the claim is established.

Example 3.12. $\vdash [\exists x \neg Gx \vee \forall x Gx]$

The beginning of the proof should be obvious:

$$
\begin{array}{c}
\neg [\exists x \neg Gx \vee \forall x Gx] \\
\vdots \\
\neg \exists x \neg Gx \\
\neg \forall x Gx \\
\vdots \\
\forall x \neg \neg Gx \\
\vdots \\
\exists x \neg Gx
\end{array}
$$

At this point one cannot proceed with '$\forall x \neg \neg Gx$' and add, say, '$\neg \neg Gc$', because no individual constant has appeared so far on the branch. Therefore I continue with '$\exists x \neg Gx$' and turn then to '$\forall x \neg \neg Gx$':

$$
\begin{array}{c}
\neg [\exists x \neg Gx \vee \forall x Gx] \\
\vdots \\
\neg \exists x \neg Gx \\
\neg \forall x Gx \\
\vdots \\
\forall x \neg \neg Gx
\end{array}
$$

$$
\begin{array}{c}
| \\
\exists x \neg Gx \\
| \\
\neg Gc \\
| \\
\neg\neg Gc \\
| \\
\underline{Gc}
\end{array}
$$

In the following example we need to use a formula twice. This is also the reason why the tableau system does not provide a decision procedure for predicate logic. Even if ticked formulae as in propositional logic, we could not be sure that the tree cannot be closed even if we had ricked all formulae, because it might be necessary to use a formula twice.

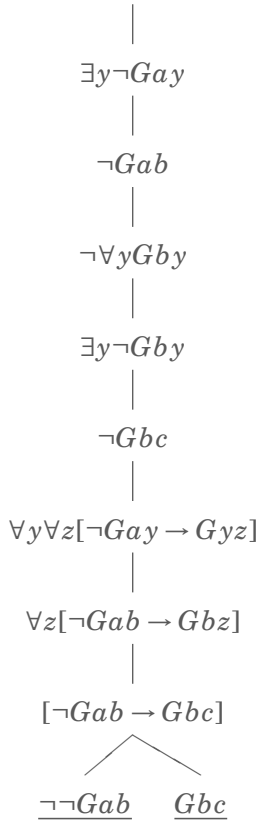
Example 3.13. $\forall x \forall y \forall z[\neg Gxy \rightarrow Gyz], a = a \vdash \exists x \forall y Gxy$

First we start the proof in a straightforward way:

$$
\begin{array}{c}
\forall x \forall y \forall z[\neg Gxy \rightarrow Gyz] \\
a = a \\
\neg \exists x \forall y Gxy \\
| \\
\forall x \neg \forall y Gxy \\
| \\
\neg \forall y Gay \\
| \\
\exists y \neg Gay \\
| \\
\neg Gab
\end{array}
$$

But now we go back to $\forall x \neg \forall y Gxy$ and replace x by b this time. The rest of the proof should not be surprising.

$$
\begin{array}{c}
\forall x \forall y \forall z[\neg Gxy \rightarrow Gyz] \\
a = a \\
\neg \exists x \forall y Gxy \\
| \\
\forall x \neg \forall y Gxy \\
| \\
\neg \forall y Gay
\end{array}
$$



3.4 Interpretations and counterexamples

In propositional logic the tableau method yields a decision procedure for deciding whether a syntactic sequent is correct or not, that is, for deciding whether a formula is provable from a finite set of formulae.[7] If all complex formulae are ticked and the tableau is not yet closed, we have done everything we can do. Therefore the sequent cannot shown to be correct in this case. And from Theorem 2.17 we know that the corresponding semantic sequent is also not correct.

If a tree does not close in predicate logic, that doesn't show anything (apart from some special cases). It could still close within the next 100 steps. The reason is that we could still try another individual constant.

For predicate logic Hodges does not define correctness for semantic sequents, that is, '$X \vDash \phi$' is not defined where ϕ and all elements of X are sentences of $\mathscr{L}_2$.[8] Counterexamples appeal to validity of arguments in English. The idea is that a syntactic sequent '$X \vdash \phi$' (X a set of formulae of $\mathscr{L}_2$,

[7] Here it is important that Hodges allows only finitely many premisses in a sequent.

[8] I have given the basic definitions in section 3.5, but this material does not form part of the syllabus.

ϕ a formula of $\mathscr{L}_2$) is not correct if there is a translation of all the formulae into English such that the resulting argument in English is not valid.

In Hodges' terminology a (predicate) interpretation is a translation of some predicate letters and individual constants into corresponding expressions in English. Thus, e.g., a binary predicate letter will be translated as binary predicate expressions of English (and similarly for predicate letters of arbitrary arity), and individual constants will be translated as designators of English. An interpretation is an interpretation for a particular formula if and only if the interpretation comprises translations for all predicate letters and individual constants occurring in the formula. H187

An interpretation specifies translations for the predicate letters and individual constants, but not for the identity symbol because the identity symbol is always translated in the usual way:

$$x = y \quad : \quad x \text{ is (identical with) } y$$

I give an example of an interpretation.

Example 3.14. The following is an interpretation for the formula $\forall x[Px \rightarrow Rxa]$:

Px	:	x is a tree on Christ Church meadow
Rxy	:	x is in y
a	:	Oxford

This example shows that a predicate letter might receive a fairly complex predicate expression of English as translation.

Given the above interpretation one can translate the entire formula '$\forall x[Px \rightarrow Rxa]$' into English as

All trees on Christ Church meadow are in Oxford.

Thus under the interpretation of Example 3.14 the formula '$\forall x[Px \rightarrow Rxa]$' becomes a true English sentence. It is not hard to think of an interpretation that renders '$\forall x[Px \rightarrow Rxa]$' a false sentence.

A domain is some arbitrary set. An English sentence is true in a domain, if and only if the sentence is true if only elements in the domain are considered. For instance, in the domain $\{2,4\}$ the sentence 'All numbers are even' is true. If the domain is the set of all integers the sentence is false. However, we must presuppose that the English sentence contains only designators designating objects in the domain. For instance, you may not specify 'Brazil' as the translation of the individual constant c and then take the set of all European countries as the domain of quantification. Thus the objects denoted by the translations of the individual constants must be in the domain of quantification. In the following it is always assumed that the interpretations respect this restriction.

In order to refute $X \vdash \phi$, one can specify a domain $\mathscr{D}$ plus an interpretation for ϕ and all formulae in X, such that all formulae in X are true under this interpretation in $\mathscr{D}$ and ϕ is false under this interpretation in $\mathscr{D}$.

> A counterexample to the claim that $X \vdash \phi$ (or, for short, a counterexample to '$X \vdash \phi$') consists in a specification of the following:
>
> - a domain of quantification (any set is allowed as domain of quantification; in particular the domain can be the empty set)
> - translations of all predicate letters involved in the argument to English predicates of the same arity.
> - translations of all individual constants involved in the argument (the translations must designate only objects in $\mathscr{D}$)
>
> where all formulae in X are true under this interpretation in the domain and ϕ is false under this interpretation in the domain.

Of course some sort of justification is required for the claim that a counterexample can be used to refute a claim like $X \vdash \phi$. The claim cannot be proved formally because it involves informal notions like the truth of English sentences in a given domain. But an inspection of the tableau system should show that the if all the translations of elements of X are true, then ϕ cannot be false, if $X \vdash \phi$.

Example 3.15. The following is a counterexample to '$\forall x \exists y Gxy \vdash \exists y \forall x Gxy$':

Gxy	:	x is separated by the sea from y
domain:	:	{Canada, U.S.A., Germany, France}

The translation of the premiss '$\forall x \exists y Gxy$' is then 'Everything is separated by the sea from something', which is true if we focus on the countries in the domain: Canada is separated by the sea, e.g., from France. The translation of the conclusion '$\exists y \forall x Gxy$' is 'Something is separated by the sea from everything.' is clearly false, because, e.g., France is not separated from Germany by the sea.

There are also simpler counterexamples like the following:

Gxy	:	x is identical to y
domain:	:	$\{1,2\}$

This is a counterexample because 1 and 2 are both identical to something in the domain, respectively (namely to themselves); but neither 1 nor 2 are identical to everything in the domain.

If there is a counterexample to a sequence, then the sequence is not correct (i.e., there is not a corresponding closed tableau).[9] This gives a method for refuting that $X \vdash \phi$. If one gives a counterexample to the argument with all the elements of X as premisses and ϕ as conclusion, then $X \vdash \phi$ is refuted.

This means in practice that if one wonders whether an argument is valid in predicate logic, one can try to prove $X \vdash \phi$ by the tableau method in order to prove that the argument is valid or one can try to find a counterexample in order to show that the argument is not valid.

Example 3.16. There is a counterexample to

$$\forall x[Px \rightarrow \exists y Rxy], \forall x[Rxa \rightarrow Qx] \vdash \forall x[Px \rightarrow Qx].$$

Px	:	is a European capital city
Rxy	:	x is the capital of y
a	:	Italy
Qx	:	is in Italy
domain:	:	the set of all capital cities and all European countries

The translations of the two premisses and the conclusion are then:

1. Every European capital is the capital of something.

2. Every capital of Italy is in Italy.

3. Every European capital is in Italy.

Sometimes it may be easier to specify a structure rather than a translation in order to refute the validity of an argument. I shall explain this in some detail in section 3.5. For those who do not want to read the next section (because it does not belong to the syllabus), I give here only an example, because it is permissible to use this approach instead of the above translation based approach.

Basically one assigns a relation in the set-theoretic (see 9) to each predicate letter occurring in the sequent. That is, instead of giving a translation for each predicate letter, one specifies a n-place relation for every n-place predicate letter. Basically the idea is this. Instead of providing a translation for each predicate letter, one assigns a truth value to each 0-place predicate letter, a set of objects to each 1-place predicate letter, and a set of ordered n-tuples to each n-place predicate letter, where $n > 1$.

In the case of Example 3.16 one could write:

[9] This cannot be proved in the formal sense because the notion of a counterexample is not precise. In the next section, however, the informal notion of a counterexample is replaced by the mathematically precise notion of a countermodel and then one can prove that an argument in predicate logic is provable if and only if there is no countermodel.

P	:	{Rome, Berlin}
R	:	{⟨Rome, Italy⟩}, {⟨Berlin, Germany⟩}
a	:	Italy
Q	:	{Rome}
domain:	:	{Rome, Berlin, Italy, Germany}

All the objects (Rome, Berlin, Italy, Germany) in the relations must be in the domain. The structure follows the idea of the above translation with the restriction to two capitals and two countries.

This approach is allowed and sometimes sensible. But only in section 3.5 I shall show how an assignment of relations to predicate letter determines the truth an falsity of a closed formula.

It is possible to assign a predicate letter the empty relation (i.e., the empty set). It is also possible to assign the same relation to two different predicate letters of the same arity.

Note. In counterexamples you should not presuppose any particular knowledge only you have. For instance, taking the set of the objects on your desk as domain is not a good idea. It is unlikely that the examiner knows the objects on your desk. Usually it is also sensible to choose a simple counterexample if possible.

3.5 The semantics of predicate logic

This section does not form part of the syllabus, because it is relatively technical and advanced. But it is a philosophically important part of logic and it clarifies the section on counterexamples.

Appendix A: Rules for dropping brackets

Most logicians employ certain rules for dropping brackets. For instance, they do not write '$[[P \wedge Q] \wedge R]$', but rather simply '$P \wedge Q \wedge R$'. The conventions explained here do not form part of the syllabus, but they should be useful when reading texts not following Hodges' somewhat idiosyncratic notation.

It is not recommended that you use these conventions. The bracketing conventions introduce more possibilities for mistakes. If brackets are missing from your formulae, the conventions below will be applied.

The conventions below concern the language $\mathscr{L}_1$ of propositional logic and the language $\mathscr{L}_2$ of predicate logic.

The expression that is obtained from dropping brackets is not itself a formula but rather a mere abbreviation of the original formula.

Bracketing Convention 1. *Brackets surrounding the whole formula may be dropped.*

For instance, one may write '$P \rightarrow [Q \vee P]$' instead of '$[P \rightarrow [Q \vee P]]$'. However, this convention does not allow to drop any brackets from '$\neg[P \rightarrow [Q \vee P]]$', because only a part and not the whole formula is surrounded by the brackets.

This convention is a possible source for errors. Assume the formula '$P \wedge Q$' is to be negated. Then it seems natural to write '$\neg P \wedge Q$'. But the latter formula is *not* the negation of the first. By Convention 1, '$P \wedge Q$' is short for '$[P \wedge Q]$' and thus its negation is '$\neg[P \wedge Q]$'. From the last formula no brackets can be dropped.

Bracketing Convention 2. *If '$[\phi \wedge \psi] \wedge \chi$' is a part of a formula, the occurrences of the brackets may be dropped and '$\phi \wedge \psi \wedge \chi$' may be written. An analogous convention applies to '$\vee$'.*

According to this convention one may write '$[Ga \wedge Gb \wedge Gc]$' instead of '$[[Ga \wedge Gb] \wedge Gc]$', for instance. Using the Convention 1 this may even be shortened to '$Ga \wedge Gb \wedge Gc$'. Convention 2 also allows to write '$\forall x[Px \rightarrow [Qx \vee Rx \vee Sxa]]$' instead of '$\forall x[Px \rightarrow [[Qx \vee Rx] \vee Sxa]]$'.

In mathematics one may write '$3 \cdot 5 + 4$' instead of '$(3 \cdot 5) + 4$', because '$\cdot$' binds more strongly that '$+$'. Similar conventions are adopted in logic.

We first fix which truth-functor symbol (connective) binds more strongly. A connective $\circ$ binds more strongly than another connective $\bullet$ if and only if $\circ$

stands further to the left than • in the following line:

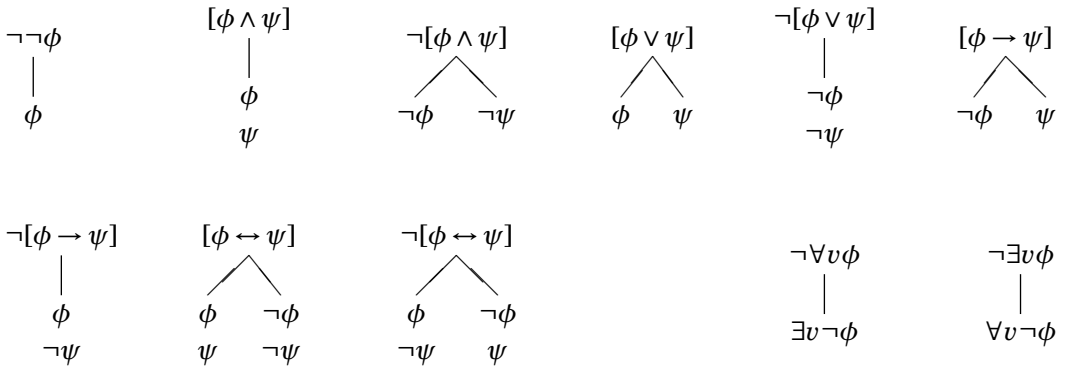
$$\wedge \quad \vee \quad \rightarrow \quad \leftrightarrow$$

The last bracketing convention allows one to drop brackets where the grouping of the symbols is already clear from the above list.

Bracketing Convention 3. *Assume* $\circ$ *and* $\bullet$ *are truth-functor symbols (connectives). If a formula contains an expression of the form* '$[\phi \circ \psi] \bullet \chi$' *(or* '$\phi \bullet [\psi \circ \chi]$'*) and* $\circ$ *binds more strongly than* $\bullet$*, one may write* '$\phi \circ \psi \bullet \chi$' *(or* '$\phi \bullet \psi \circ \chi$' *in the latter case).*

'$[P \wedge Q] \rightarrow R$' (Convention 1 has already been used) may be shortened to '$P \wedge Q \rightarrow R$' because '$\wedge$' binds more strongly than '$\rightarrow$'.

Appendix B: tableaux rules



$$\begin{array}{c}\phi\\ D = E\\ |\\ \psi\end{array}$$

provided that the individual constant D occurs in ϕ, and ψ is the result of replacing one or more occurrences of D in ϕ by occurrences of E.

$$\begin{array}{c}\phi\\ E = D\\ |\\ \psi\end{array}$$

provided that the individual constant D occurs in ϕ, and ψ is the result of replacing one or more occurrences of D in ϕ by occurrences of E.

$$\begin{array}{c}\forall v\phi\\ |\\ \psi\end{array}$$

provided that there is an individual constant D which has already occurred in the branch above ψ, and ψ is the result of replacing every free occurrence of the variable v in ϕ by D.

$$\begin{array}{c}\exists v\phi\\ |\\ \psi\end{array}$$

provided that ψ is the result of replacing every free occurrence of v in ϕ by the individual constant D and D has not occurred anywhere in the branch above ψ.

You may draw a line at the bottom of every branch on which a formula occurs together with its negation or on which '$\neg D = D$' occurs for some individual constant D.

Appendix C: Quotation

In logic one talks about expressions in natural and formal languages. By enclosing an expression in quotation marks one obtains a term designating that expression. For instance, the expression

'Italy'

refers to the word that begins with an 'I' followed by 't', 'a', 'l' and 'y'.

When expressions are displayed, quotation marks are usually skipped.

Often in logic one does not only intend to talk about a single expression, but about many expressions of a certain form simultaneously. For instance, one tries to affirm the followiong claim:

> A conjunction of two sentences is true if and only if both sentences are true.

This can be more formally expressed in the following way:

(K1) If ϕ and ψ are sentences, then the expression beginning with ϕ followed by '$\wedge$' and ψ is true if and only if ϕ and ψ are true .

Here we do not talk about 'ϕ' and 'ψ', rather 'ϕ' and 'ψ' are used as variables ranging over sentences. These variables belong to the language we are actually using, that is, in English enriched by some symbols. Therefore there is no need for quotation marks enclosing 'ϕ' and 'ψ', respectively. But we talk about the conjunction symbol '$\wedge$', so it has to be enclosed in quotation marks.

We adopt a convention for abbreviating claims like (K1). According to this convention (K1) can be rephrased in the following way:

(K2) If ϕ and ψ are sentences, then '$\phi \wedge \psi$' is true if and only if ϕ and ψ are true .

We still do not intend to talk about the Greek letters 'ϕ' and 'ψ', but rather about sentences of a language that obtained by replacing sentences for the Greek letters 'ϕ' and 'ψ' in '$\phi \wedge \psi$'.

The convention has been introduced here only by way of example, and this should be sufficient for the understanding of the text here and Hodges' book. The details of this convention are tricky and quotation has puzzled logicians and philosophers. There are ways to avoid quotation marks altogether, but this is usually on he cost of readability.

BIBLIOGRAPHY

Hodges, Wilfrid (2001), *Logic: An Introduction to Elementary Logic*, second edn, Penguin Books, London.