KAI BÖRGE HANSEN

APPLIED LOGIC

UPPSALA 1996
Abstract


This collection of articles and essays contains contributions to foundational studies and to pure and applied logic.

In Chapter 2, it is proved that it is formalisation itself which gives rise to the paradoxes of implication. Therefore no adequate formal logic of conditionals is possible. Foundations for an informal logic are suggested.

Chapter 3 is concerned with the nature of philosophy. An analysis of the logical structure of theories leads to a theory of philosophy closely related to the original Pythagorean conception. In Chapter 4, I show that classical rationalism need not be considered refuted. By combining some of Descartes’s ideas with concepts from modern logic, it appears that the synthetic a priori is the logic of self-referential systems.

The next chapters are concerned with mathematical logic. In Chapter 5, the usual derivation of Russell’s antinomy is shown to be an application of Cantor’s diagonal procedure. Chapter 6 contains a paradox based on the upward Löwenheim-Skolem theorem, in analogy with Skolem’s paradox based on the downward Löwenheim-Skolem theorem. Chapter 7 contains results and open problems on co-consistency. Chapter 8 gives new or partly new proofs of some well-known theorems in logic and geometry.

In Chapter 9, the logical foundations of logic programming are developed. This is done from a logician’s rather than from a computer scientist’s point of view.

Key words: Logic, conditionals, metaphilosophy, synthetic a priori, Russell, Skolem, co-consistency, logic programming, Prolog.

Kaj Berge Hansen, Department of Philosophy, Vilhelmso 3, S-752 36 Uppsala, Sweden

Dedicated to the memory of my parents

KAREN (1916-1971)
BØRGE (1910-1992)
Acknowledgments

The articles and essays in the present collection were written between 1994 and 1996. They are based on material developed during the last two decades.

I am indebted to Paullia for moral support during the work on this book.

I am grateful to head librarian Thomas Tronc, Uppsala University Library, and Professor Lennart Elvins, Uppsala University, for their help in making the publication of the book in the series Acta Universitatis Upsaliensis possible.

Uppsala in June 1996

Kay Bengt Hansson
Contents

1. Introduction ........................................ 11
   1-1 Credo ........................................... 11
   1-2 Summary ....................................... 12

2. Conditionals and the Foundations of Logic ........... 14
   2-1 The Idea of Formalisation ...................... 14
   2-2 The Paradoxes of Implication .................. 15
   2-3 Object Language and Metalanguage ........... 17
   2-4 Truth-Tables .................................... 18
   2-5 Other Preliminary Remarks ...................... 20
   2-6 'If-then' and '→' are NOT Equivalent .......... 23
   2-7 'If-then' and '→' ARE Equivalent .............. 25
   2-8 Analysis ......................................... 28
   2-9 On Formal Logic ................................ 32
   2-10 Foundations of Logic .......................... 33
   2-11 Importance of the Problem .................... 39
   2-12 Other Solutions ................................ 44
   2-13 Summary and Conclusion ........................ 49

3. What is Philosophy? ................................... 53
   3-1 Introduction .................................... 53
   3-2 Arithmetical Theories ........................... 54
   3-3 Other Theories ................................... 60
   3-4 Philosophy ....................................... 72

4. Logical Rationalism: A Programme ..................... 83
   4-1 Introduction .................................... 83
1. Introduction

1.1 Credo

1.1 A theory is defined by a language, a logic, and a set of nonlogical axioms.

1.2 A theory consists of foundations and superstructure. To the foundations belong the basic principles of the theory, e.g., logical principles, methodological principles, mathematical axioms, and laws of nature.

Foundations are important because of what can be based upon them, namely the superstructure. The superstructure consists of the logical consequences of the basic principles incorporated in the foundations.

1.3 Logic is an essential part of the foundations of all theories. For some theories, e.g., mathematics, quantum theory, and computer science, there are still open logical problems in the foundations.

The development of the superstructure consists in deriving logical consequences from the foundations. The theoretical development of any science, for fixed foundations, belongs to applied logic.

1.4 We may develop a theory by developing its foundations or its superstructure. If a problem can be solved on the basis of the given foundations, its solution is only a question of applied logic. If the problem is unsolvable on the given foundations, these must first be properly developed. On the basis of the revised foundations, the problem can now be solved by logic. We have the identity

\[ \text{problem solving} = \text{foundations} + \text{logic} \]
1-2 Summary

2.1 This book contains contributions to foundational studies and to pure and applied logic. Problems in logic, philosophy, mathematics, and logic programming are treated.

2.2 Foundations of Logic. Chapter 2, Conditionals and the Foundations of Logic, takes the well-known paradoxes of implication as starting point. It is shown that they are inevitable consequences of formalisation. Therefore the problem of finding a satisfactory conditional logic cannot be solved in any formal system. New foundations for an informal logic based on functional connections at the object level rather than on truth functional connections at the object language level are suggested.

2.3 Philosophy. Chapter 3 is devoted to the old question about the nature of philosophy. The basic idea is that philosophy is thinking and that therefore a study of the logical structure of theories may provide insights into the nature of philosophy. Thus this chapter is an investigation into the logical foundations of philosophy. Gödel's incompleteness theorems lead to a theory of philosophy as foundational studies. This conception of philosophy is arguably closely related to the original Pythagorean conception.

In the 17th and 18th centuries, rationalism and empiricism were the two dominant movements in philosophy. After Kant, rationalism declined in appeal and influence while empiricism kept flourishing. When at the beginning of this century empiricism was amalgamated with the new formal logic, the strongest and most influential form of empiricism ever was created. In Chapter 4, I show that classical rationalism contains ideas which have not yet been properly exploited. Using concepts and results from modern logic, I show that the synthetic a priori may be seen as the logic of self-referential systems and models. Thus an amalgamation of mathematical logic and some ideas from classical rationalism may give rise to a new form of rationalism.

2.4 Mathematics. In Chapter 5, it is shown how the usual derivation of Russell's antinomy may be seen as an application of Cantor's diagonal procedure.

In Chapter 6, a logical analysis of Skolem's paradox is given. An omission in Skolem's and later logicians' analysis of the paradox is detected and corrected. Skolem's paradox is based on the downward Löwenheim-Skolem theorem. An analogous paradox based on the upward Löwenheim-Skolem theorem is formulated and solved. An application to Lindström's theorem is suggested.

The concept of Ω-consistency was introduced by Gödel in his famous paper on the incompleteness of arithmetic. Ω-consistency is, however, of considerable interest in its own right. Chapter 7 contains results and open problems on Ω-consistency.

In Chapter 8, apparently new proofs, or proofs which seem at least to contain some new details, are given of a number of known theorems. Some new examples and counterexamples are also included. The fields of logic covered are Padoa's method, decidability and undecidability, recursion theory, arithmetic theories, and quantum logic. The last section contains a proof of the circle of Apollonius.

2.5 Logic Programming. Chapter 9 is a survey article on the logical foundations of logic programming, especially programming in Prolog. The whole chapter is written from a logician's rather than from a computer scientist's point of view. The starting point is the standard form of predicate logic. General clauses and Horn clauses are defined. The transformation from standard form to clause form is treated as are the resolution, unification, and contraction rules, the three deduction rules of clause logic. Section 5 indicates the road from Horn clause logic to Prolog. In Section 6, a number of completeness theorems for clause logics and Prolog are proved. Thus it is proved that general clause logic, Horn clause logic, and Prolog are complete relative to the usual set theoretic semantics of predicate logic. A new proof is given of the computability of all recursive functions in Horn clause logic and Prolog. The last section contains model theoretic results on Horn formulas. One consequence of the main theorem in this section is that there are problems which cannot be formulated and solved in Horn clause logic and Prolog by direct and efficient methods.
2. Conditionals and the Foundations of Logic

2-1 The Idea of Formalisation

1.1 It is generally agreed that the birth of modern logic coincides with the publication of Frege's book *Begriffsschrift* in 1879. A fundamental idea in Frege's work, and in practically all later work in logic, is that logic is a formal science. It is hardly an exaggeration to say that the idea of formalisation is by far the most important and influential idea in the whole scheme of modern logic.

1.2 The basic principle of formal logic is that logical truth and logical validity depends only on the logical form of sentences and not on the meaning (the content) of their nonlogical components. To every declarative sentence, we can assign a formula which represents its logical form. Formulas are concrete and precisely defined objects. It is possible by an algorithm (and therefore intersubjectively) to decide whether a given string of symbols is or is not a formula of a given recursive language.

Moreover, the logical relations between formulas are defined by a recursive set of deduction rules and logical axioms. A consequence of this is that the proof relation

\[(C_1, \ldots, C_p) \text{ is a proof in logic } L \text{ of } B \text{ from the premises } A_1, \ldots, A_n\]

is a recursive relation.

1.3 Consequences. I mention some of the beneficial consequences of formalisation:

(I) Since the proof relation is recursive, it can always be objectively and intersubjectively decided whether a proposed proof really is a valid proof. Logic becomes a science in the strict sense.

(II) Formalisation and the semantics of the formal logic makes it possible to remove all ambiguities from logic. Formalisation has made logic an exact science.

(III) The basic scheme of a formal logic can easily be varied by taking different sets of logical operators, logical axioms, and deduction rules. This has led to the discovery of a large number of alternative logics like, e.g., classical first order logic, higher order logics, co-logic, type theory, intuitionistic logic, quantum logic, intermediate logics, modal logics, nonmonotonic logic, etc. Formalisation has promoted creativity in logic.

(IV) Formal logical systems and formalised theories are well-defined mathematical structures which can be studied by mathematical methods much the same way different geometries, number systems, and algebras can be thus studied. Formalisation has given rise to mathematical logic.

(V) Since the proof relation is recursive, logical operations can in a formal logic be performed by a computer. The computer can search for proofs. This makes logic programming, e.g., in Prolog, possible. Formalisation has made the applications of logic to logic programming and artificial intelligence possible.

1.4 We see that the blessings which formalisation has bestowed upon mankind are numerous and great. Few logicians nowadays can imagine any other kind of logic than formal logic.

2-2 The Paradoxes of Implication

2.1 The Paradoxes. There are a number of well-known paradoxes of implication in classical logic, e.g.,

\[\vdash B \rightarrow (A \rightarrow B)\]

\[\vdash \neg A \rightarrow (A \rightarrow B)\]

\[\vdash (A \rightarrow B) \lor (B \rightarrow A)\]

\[\vdash (A \rightarrow B) \lor (\neg A \rightarrow B)\]
\[ (A \rightarrow B) \lor (A \rightarrow \neg B) \]
\[ (A \rightarrow B) \lor (B \rightarrow C) \]

2.2 The Problem of Conditionals. Many people feel that the formulas in § 2.1 are counterintuitive. The problem of developing a theory of conditionals, which is in better agreement with our intuition than classical logic is, is known as the problem of conditionals.

To some, this is a problem of philosophy of language which does not affect the foundations of logic. To others, including myself, the problem of conditionals is a central problem in the foundations of logic.

There have been very many attempts this century at solving the problem of conditionals and the closely related problem of entailment. They share two characteristics:
1. Each attempt consists in devising an alternative formal system.
2. They have failed in solving the problem.

2.3 PROBLEM. In this essay, I will prove that it is formalisation itself which is the cause of the implication paradoxes. Therefore the problem of conditionals cannot be solved in any formal system. I also sketch some ideas for an informal conditional logic.

2.4 Organisation. The essay is organised as follows. In Sections 3–5, I outline simple preliminaries for the analysis. In Section 6, it is shown by an example that material implication and 'if-then' are not equivalent. In the next section, I prove informally that they are equivalent. Then in Section 8, this antinomy is analysed and solved. In the next two sections, it is proved that the problem of conditionals cannot be solved in any formal logic. The consequences of this for the foundations of logic are discussed. In Section 11, the importance of the problem of conditionals is examined. This analysis gives some support to the idea of an informal logic. Finally, in Section 12, some celebrated solutions to the problems of conditionals and entailment are examined and criticised.

2-3 Object Language and Metalanguage

3.1 The distinction between object, object language, and metalanguage is well-known. It is explained by some examples:

At the object level: Objects, e.g.,
- snow
- Tartu

Properties of objects, e.g.,
- (being) white
- (being) a university town

States of affairs, facts, e.g.,
- that snow is white
- that Tartu is a university town

At the object language level: Sentences on states of affairs at the object level, e.g.,
- Snow is white
- Tartu is a university town

Properties of such sentences, e.g.,
- (being) true
- (being) false

At the metalanguage level: Sentences on states of affairs at the object language level, e.g.,
- 'Snow is white' is true
- 'Tartu is a university town' is true

3.2 The T-Principle. There is a simple principle which allows us to move between the different language levels. This is the T-principle (sometimes called the equivalence principle):

For any sentence 'A',
'A' is true iff A

3.3 EXAMPLES. Here are two concrete examples of the T-principle:

'Snow is white' is true iff snow is (as a matter of fact) white
'Tartu is a university town' is true iff Tartu is a university town
3.4 REMARK. It is of some importance to realize that the equivalence

'Snow is white' is true iff snow is white

is not a logical equivalence. For the left-hand side implies that there exists a sentence, namely 'snow is white', while the right-hand side does not have this implication. I have stated the T-principle only for given sentences A. With this proviso, it seems to be valid.

2-4 Truth Tables

4.1 Truth Tables. The truth tables for ¬, ∧, ∨, and → are well known:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>¬A</th>
<th>A ∧ B</th>
<th>A ∨ B</th>
<th>A → B</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

4.2 COMMENT. The truth-tables for ¬, ∧, ∨ may seem natural. Normally, they are accepted without protests.

The table for → is somewhat strange if it is supposed to represent 'if-then'. During classes in elementary logic, it is almost always met with protests from some of the students. Line 2 may seem OK. When A is true and B is false, then

If A then B

and therefore

A → B

should certainly be false. But to write T in the other three lines seems completely arbitrary. Why not F instead? Even line 2 is suspect. If A then B may be false also in other cases than when A is true and B is false.

Indeed, the truth-table for → seems to have almost nothing with the usual conditional of natural languages to do. The implication paradoxes are immediate consequences of the truth-tables.

4.3 EXAMPLE. The following three conditionals are justified by lines 1, 3, 4 in the truth-table. When they are read as natural conditionals, they run counter to our linguistic and logical intuition.

Napoleon lost the battle at Waterloo → snow is white
Napoleon won the battle at Waterloo → snow is white
Napoleon won the battle at Waterloo → snow is black

4.4 REMARK. Note that truth-functions are entities at the object language level. Truth-tables, and therefore truth functions, are defined in the metalanguage. Therefore what these definitions are about, namely the truth functions, must belong to the object language level.

4.5 Truth Conditions. It is possible to express the truth conditions for ¬, ∧, ∨, → as equivalences where on the left-hand side we have a formal connective and on the right-hand side we have the corresponding informal connective:

(4-1) ¬A is true ⇔ A is not true
(4-2) A ∧ B is true ⇔ A is true and B is true
(4-3) A ∨ B is true ⇔ A is true or B is true
(4-4) A → B is true ⇔ if A is true, then B is true

These equivalences show that there is a very close kinship between the formal connectives and their informal counterparts.

Again the equivalences for ¬, ∧, ∨ may seem natural and can be read off directly from the truth-tables. E.g., from the truth-table for ∧ we see immediately that

A ∧ B is true ⇔ A is true and B is true

On the other hand, the equivalence for → does not seem to follow trivially, even if we accept the truth-table for →. In Section 7, a proof of this equivalence will be given.

4.6 We see that the relation between 'if-then' and '→' is somewhat problematic. They seem not to be equivalent. '→' seems not to be a good representative for 'if-then'.

4.7 REMARK. Note that in equivalences (4-1)-(4-4), the formal connectives on the left-hand sides belong to the object-language level while
the corresponding informal connectives on the right-hand sides belong to the metalanguage. It will turn out that this difference is crucial.

2-5 Other Preliminary Remarks

5.1 Criteria. We would normally expect that a good logical system or logical theory L should satisfy the following criteria.
(1) Soundness. L must never take us from true premises to a false conclusion.
(2) Completeness. For every intuitively correct inference, it must be possible to represent it and perform it in L.
(3) Correspondence. The logical constants of L should be good (faithful) representatives of their natural counterparts. (This is of some importance for applications.)

5.2 Ordinary classical first-order logic CL can be proved to satisfy the soundness criterion. For all we know, CL satisfies the completeness criterion.

CL does not to 100% satisfy the correspondence criterion; but experience shows that CL is quite close to satisfying even this third criterion. With a little cautiousness, we can formalise natural language sentences in sentential or predicate logic in a straightforward way and with very good results. In Section 2, I will prove a theorem which explains why CL works so relatively well even in this respect.

5.3 EXAMPLE. We now consider three examples of undoubtedly correct uses of conditionals.

Suppose we have a coffee automaton from which it is possible for 5 kroner to buy a mug of coffee. This can be expressed by a conditional

(5-1) IF you insert 5 kroner and you press button B,
THEN you'll get a mug of coffee with milk and without sugar.

The two events in the antecedent (input) are causes of the event in the consequent (output).

Causal connections can be expressed by conditionals.

Note that the conditional also may be interpreted to express a functional connection. Every automaton is a function which takes an input and an inner state to an output.

5.4 EXAMPLE. Consider a function, e.g.,

\[ y = f(x) = x^2 \]

Given this function, we can form conditionals, e.g.,

(5-2) IF \( x = 2 \), THEN \( y = 4 \)

Functional connections can be expressed by conditionals. If the function is a constant function, the use of the conditional may be doubtful; otherwise not.

5.5 EXAMPLE. For an analogy to be used later, we also consider the function

\[ z = f(x,y) = x^2 + y^2 \]

The graph of this function is an elliptic paraboloid. We now take a subset \( S \) of this graph, namely all those points for which \( z = 1 \):

\[ S = \{(x,y,1) \mid 1 = x^2 + y^2 \} \]

Given that we stay within \( S \), this can be used to form conditionals like

(5-3) IF \( x = 1 \), THEN \( y = 0 \)
(5-4) IF \( x = 0 \), THEN \( y = +1 \) or \( y = -1 \)

Each conditional expresses a functional connection between antecedent and consequent. If the functional connection is a constant function or if the consequent is an uninformative disjunction of all possibilities, the use of the conditional may be doubtful. Otherwise it seems justified. This use of conditionals is well established among mathematicians.

5.6 DISTINCTION. We will also need a distinction between information based conditionals and reality based conditionals. This distinction will be explained by two examples.

5.7 EXAMPLE. We have two objects a and b and a box W. I show you that the following are all possible:

(1) a is in W, b is in W;
(2) a is in W, b is not in W;
(3) a is not in W, b is in W;
(4) a is not in W, b is not in W.

We use the following sentential parameters:

\[ A: \text{a is in W} \]
\[ B: \text{b is in W} \]

I ask you to close your eyes, and I place a in W and b outside W. I then give you the piece of information that

\[ a \text{ and } b \text{ are not both in W} \]

i.e.,

\[ \neg(A \land B) \]

I ask you:

If a is in W, then what about b?

You can infer:

(5-5) If a is in W, then b is not in W,

i.e.,

\[ A \rightarrow \neg B \]

This is an information based conditional. It is based on the piece of information you get from me and only on that. From the exhaustive set of possibilities (1)-(4) above we see that nothing in reality forces this conditional to be true. It does not express a functional relation between a being in W and b being in W. It does express a functional relation between the piece of information that a is in W being given or true and the piece of information that b is in W being given or true. Thus this functional relation belongs to the information level, not to the object level (or reality level).

The conditional may be seen as a statement about information sets. We start with the information set

\[ K = \{ \text{a and b are not both in W} \} = \{ \neg(A \land B) \} \]

The content of the conditional is the following:

If K is expanded with the piece of information

\[ a \text{ is in W} \]

then the expanded set \( K^* = \{ \neg(A \land B), A \} \) of information also contains the piece of information \( \neg B \); i.e.,

\[ \neg B \text{ is not in W} \]

while K does not.

Thus the functional relation on which the conditional is based is in this case a relation between the sets of information K and K*.

5.8 EXAMPLE. We have two congruent cubes a and b and a cubic box W. W has a size so that it just has space for one of a and b, but not both at the same time. Now you can infer

(5-7) If a is in W, then b is not in W

i.e.,

(5-8) \[ A \rightarrow \neg B \]

This conditional is reality based. It is based on the impossibility, obtaining in reality, of having both a and b in W at the same time.

In general, all conditionals which are based on laws of nature are reality based. Even the conditionals considered in examples 5.3-5.5 may be considered reality based. In each of the cases, the conditional is based on a functional connection at the object level, in reality.

5.9 OBSERVATION. For an information based conditional it may occur that if we get more information, then the functional connection (between pieces of information or sets of information) on which the conditional is based may disappear so that the conditional is no longer warranted. Thus if \( K_1 = \{ \neg(A \land B), \neg B \} \), \( K_1 \) cannot be used as basis for the conditional \( A \rightarrow \neg B \). A reality based conditional, on the other hand, is stable and unshaken no matter how much information we get. This gives a criterion for distinguishing between information based conditionals and reality based conditionals.

2.6 'If-then' and '→' are NOT Equivalent

6.1 I show by an example that '→' does not exactly and adequately represent 'if-then'. I give the best example I know of.
6.2 EXAMPLE (After Cooper (1968)). We consider the following statement.

If I am in Odense then I am in Denmark, and if I am in Tartu then I am in Estonia. This implies that either it is the case that if I am in Odense then I am in Estonia, or else it is the case that if I am in Tartu then I am in Denmark.

Using the sentential parameters

O: I am in Odense
T: I am in Tartu

D: I am in Denmark
E: I am in Estonia

the statement can be formalised as

\[(O \rightarrow D) \land (T \rightarrow E) \rightarrow [(D \rightarrow E) \lor (T \rightarrow D)]\]

This is a valid logical consequence. 

Interpretation (I): Interpret \(\rightarrow\) in formula (6-1) as truth-functional implication. Then the formula is logically true.

Interpretation (II): Now interpret the arrow in formula (6-1) as if-then, otherwise leaving the interpretation of the formula unchanged. This makes the formula, and the original statement, false.

6.3 CONCLUSION. This example shows beyond doubt that \(\rightarrow\) does not exactly represent 'if-then'. Somewhere in the development of the foundations of formal logic something has gone wrong.

In the next section, I prove that \(\rightarrow\) and 'if-then' are equivalent which contradicts the result of the present section. The analysis of this paradox turns out to throw considerable light on what is wrong in the foundations of logic.

6.4 REMARK. Note that Example 6.2 shows that \(A \rightarrow B\) and 'if A then B' have different truth conditions. The conditionals we study are based on functional connections between A and B. The existence of the functional connection is part of the truth condition for 'if A then B'. On the other hand, 'A \rightarrow B' may be true without any functional connection between A and B. Therefore the truth conditions are different. Example 6.2 brings this out nicely.

Some philosophers of language have claimed that \(A \rightarrow B\) and 'if A then B' have the same truth conditions but different acceptability conditions (see Jackson (1987) or Read (1994)). This contradicts our result in §§ 6.2-6.3. It is this divergence in truth conditions which makes the analysis of conditionals interesting for logicians, not just a matter for linguists and philosophers of language.

6.5 REMARK. Cooper (1968) contains several nice examples similar to Example 6.2.

2-7 'If-then' and '→' ARE Equivalent

7.1 In this section, I give an informal proof which apparently shows that 'if-then' and '→' are equivalent.

7.2 PROPOSITION (Truth condition for \(\rightarrow\)).

A \(\rightarrow B\) is true \iff if A is true then B is true

PROOF:

Assume that A \(\rightarrow B\) is true. Then we must keep to the truth-values T of A \(\rightarrow B\) marked by bold and double under-lining in the truth-table for \(\rightarrow\) (which is here written in matrix form).

We ask

If A is true, then what about B?

Following the arrows, we see that

\[(7-1) \quad \text{If A is true, then B is true.}\]

We give an indirect proof. Assume

\[(7-2) \quad \text{If A is true, then B is true}\]

and
(7-3) \( A \rightarrow B \) is false

From (7-3) and the truth-table for \( A \rightarrow B \), we get

(7-4) \( A \) is true

(7-5) \( B \) is false

(7-2) and (7-4) give

(7-6) \( B \) is true

which contradicts (7-5).

7.3 COMMENT. In the proof for the direction \( \Rightarrow \), we may see the truth-table for \( \rightarrow \) as an automaton. The assumption that \( A \rightarrow B \) is true determines an inner state in the automaton, indicated by bold and double underlining. There are four possible inputs which may be given to the automaton:

- \( A \)-true, \( B \)-false
- \( A \)-false, \( B \)-true
- \( A \)-false, \( B \)-false
- \( A \)-true, \( B \)-true

Moreover, it is the input \( A \)-true which makes \( B \) true; because for the alternative \( A \)-input, \( A \)-false, the truth value of \( B \) is indeterminate. Therefore the use of 'if-then' in the proof is justified and uncontroversial (cf. examples 5.4 and 5.5.)

7.4 COMMENT. The use of if-then in the proof is completely analogous with Example 5.5. The truth-table for \( \rightarrow \) determines a graph consisting of the four triples

- \((T,T,T); (T,F,F); (F,T,T); (F,F,T)\)

We then select the subset \( S \) of the graph for which the value of the truth function is T:

\[ S = \{(T,T,T), (F,T,T), (F,F,T)\} \]

They are the points marked by bold and double underlining in the truth-table in the proof of Proposition 7.2. Then we use the subset \( S \) of the graph to form a conditional, exactly as in Example 5.5.

7.5 COMMENT. The following objection has been raised to the proof of direction \( \Rightarrow \) of Proposition 7.2. Assume that we know that \( A \rightarrow B \) is true because we know that \( B \) is true, i.e.,

(7-7) \( B \) is true \( \Rightarrow \) \( A \rightarrow B \) is true

If we then also, as in the proof of Proposition 7.2, have

(7-8) \( A \rightarrow B \) is true \( \Rightarrow \) if \( A \) is true then \( B \) is true

then, by the transitivity of \( \Rightarrow \), we get the paradox of implication

(7-9) \( B \) is true \( \Rightarrow \) if \( A \) is true then \( B \) is true

Since (7-7) is possible, (7-8) must be wrong.

In comments 7.3-7.4, I have tried carefully to show that (7-8) is valid. If we have the piece of information

(7-10) \( A \rightarrow B \) is true

and only that, then

(7-11) if \( A \) is true then \( B \) is true

follows. All that the objection shows is that the conditional (7-11) is information based, as defined in Section 2.5. As shown in §§ 5.6-5.9, it is characteristic of information based conditionals that a situation as in the objection can arise. If we to the piece of information (7-10) add the further piece of information \( B \) is true, then the function, on which the conditional (7-11) is based, is changed so that it no longer can justify a conditional.

This shows that \( \Rightarrow \) need not be transitive when applied to information based conditionals. The inference (7-8) is valid when we have only the piece of information \( A \rightarrow B \) is true. Transitivity for \( \Rightarrow \) would imply that even (7-9) is correct, and this need not be the case. Of course, the transitivity of \( \Rightarrow \) can be added as a condition to the logic. This is done in the usual truth-functional logic.

Sentences are encodings of information. The predicates true and false are predicates of sentences and therefore of information. A conditional like (7-11) must be based on a functional connection between sets or pieces of information. What we learn from this objection is that truth functional implications are information based.

7.6 REMARK. Note that Proposition 7.2 has a very beneficial consequence. It shows that there is a very close kinship between 'if-then' and '→'. This explains why the logic of '→' is so close to the logic of 'if-then', in spite of the apparently crazy truth-table for '→'. It justifies to some extent the use of '→' as a formal model of 'if-then'.
7.7 REMARK. Though we have proved that there is a close connection between 'if-then' and '→', we have not proved that they are equivalent. The reason is that '→' on the left-hand side of ↔ in Proposition 7.2 occurs in the object language while 'if-then' on the right-hand side of ↔ occurs in the metalanguage. The T-principle gives an instrument by means of which we can bring them at the same language level.

7.8 PROPOSITION. Assume the T-principle. Then
A → B ↔ if A, then B

PROOF:

A → B ↔ A → B is true (* the T-principle *)
↔ if A is true then B is true (* Proposition 7.2 *)
↔ if A then B (* the T-principle *)

7.9 REMARK. Proposition 7.8 contradicts the conclusion of Section 2-6. We have a paradox. In the next section, this paradox is analysed and solved.

Note that in the third line of the proof of Proposition 7.8 we have applied an assumption to the effect that conditionals are "transitive".

2-8 Analysis

8.1 In the comments following the proof of Proposition 7.2, I have defended this proof. They show that the cause of the paradox mentioned in Remark 7.9 must be found either in the T-principle or in the use we have made of it in Proposition 7.8.

8.2 The Use of the T-Principle. In the proof of Proposition 7.8, we used the T-principle in the following way.

if 'A' is true then 'B' is true

T-principle ⊢ ⊢ T-principle
A B

We first used the principle to get A → B is true from A → B. From this we inferred by Proposition 7.2 the conditional

(8-1) If A is true, then B is true

based on a functional (actually truth-functional) connection between A-true and B-true. This functional connection exists at the object language level. From the figure we see that we use the T-principle to transport this conditional down to the object level. But though there is a (truth) functional connection between A-true and B-true at the object language level, there need not be any functional connection between A and B at the object level. And such a functional connection at the object level between A and B is what is needed to justify the conditional

(8-2) If A then B

The T-principle in itself is probably correct; but transitivity cannot be applied unlimited when the T-principle is involved. If we have a chain

(8-3) If X then Y
(8-4) If Y then Z

and (8-3) or (8-4) expresses one half of the T-principle, then

(8-5) If X then Z

need not follow.

8.3 EXAMPLE. We consider a device which consists of a box with a button and a red lamp. The red lamp is on sometimes, and sometimes not. One may press the button or abstain. There is no connection between the button being pressed and the light being on.

I ask you to close your eyes and give you the following piece of true information

(8-6) It is not the case that A is true and B is false
A: The button is pressed  
B: The red lamp is on

There are then the following possibilities for A and B:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

The fourth combination

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

is excluded by the piece of information (8-6). These are precisely the lines in the truth-table where A → B is true. As in the proof of Proposition 7.2, we get the truth-functional connection

(8-7) If A is true, then B is true

By the T-principle and transitivity for conditionals, we should then get as in Proposition 7.8

(8-8) If A then B

i.e.,

(8-8'') If the button is pressed, then the light is on

But according to the construction of the device, there is no connection (at the object level) between A (that the button is pressed) and B (that the light is on). It is a mistake to transport the functional connection in (8-7) down to the object level.

8.4 REMARK. The analysis shows that the T-principle has a peculiar property:

(8-9) The T-principle is truth-preserving; but it does not preserve functional connections.

In Proposition 7.8 we saw that when the T-principle is added to a theory of conditionals with transitivity, then the conditional collapses to material implication. The observation (8-9) shows why → is truth-preserving but does not preserve functional connections.

8.5 REMARK. Every information-based conditional expressed in the object-language rests on a functional connection at the information level (e.g., at the object language level) which by an operation as in § 8.2 is transported down to the object level.

8.6 ANALYSIS. We give an analysis of Example 6.2 using the ideas and insights obtained in the present section.

We first note that the statement in the example can be simplified since we only need one of the conjuncts in the premise. Thus Formula (6-1) may be simplified to, e.g.,

(8-9) \((O \rightarrow D) \equiv [(O \rightarrow E) \lor (T \rightarrow D)]\)

If I interpret "→" as a reality-based conditional, then \((O \rightarrow D)\) is true since there is a functional relation between being in Odense and being in Denmark. On the other hand, \((O \rightarrow E)\) and \((T \rightarrow D)\) are both false in this interpretation since the same functional relation does not hold between antecedents and consequents.

If we use (8-9) interpret "→" as an information-based truth-functional conditional, then (8-9) is true as may be shown as follows. Assume \((O \rightarrow D)\). Either \(O\) is true or false. If \(O\) is true, then \(D\) is true and hence \((T \rightarrow D)\) is true. Here we have used

(8-10) \(D\) is true \(\Rightarrow (T \rightarrow D)\) is true

If \(O\) is false, then the other disjunct \((O \rightarrow E)\) is true. Here we have used

(8-11) \(O\) is false \(\Rightarrow (O \rightarrow E)\) is true

In Comment 7.5, I have showed that principles like (8-10) and (8-11) can be true of information-based conditionals only. They are certainly true of truth-functional conditionals.

It should be noted also that in the case where "→" is information based, we get conditionals like

(8-12) Information A is true \(\rightarrow\) information B is true
(8-13) Information A is given \(\rightarrow\) information B is given

To transport these conditionals down to the object level, we use, and must use, the T-principle or an analogous principle as in the analysis in § 8.2.
2-9 On Formal Logic

9.1 DEFINITION. (I) A formal logic is a set of formulas, which are intended to be interpreted as logical truths, and a set of rules of inference such that the set of formulas is closed under the rules of inference.
(II) A formal conditional logic is a formal logic with an operator \( \rightarrow \) intended to be interpreted as a conditional operator.
(III) A formal logic is intuitively complete if every intuitively correct inference can be represented and performed in the logic.

(* This definition is somewhat vague; but it suffices for our present purposes.*)

9.2 PROPOSITION. Every formal conditional logic is either intuitively incomplete or else the conditional \( \rightarrow \) is equivalent with material implication.

PROOF: This is a corollary to propositions 7.2 and 7.8. Assume that the logic is intuitively complete. Then the proofs of propositions 7.2 and 7.8 can be formalised in the conditional logic such that all occurrences of 'if-then' are formalised by \( \rightarrow \). Formalising Proposition 7.8, we get

\[ A \rightarrow B \iff A \rightarrow B \]

Thus \( \rightarrow \) and \( \iff \) are equivalent.

9.3 COMMENT. It may be objected to the proof that if we just take care that \( \iff \) is not transitive, then Proposition 7.8 cannot be proved in the conditional logic. Moreover we have detected cases of conditionals which do not allow transitivity. Therefore transitivity for conditionals is not a universally valid law and should not be included in a logic. This is correct. The trouble with this move is that most cases of conditionals do allow transitivity. There is no way to distinguish on purely formal grounds between those conditionals which allow transitivity and those which do not. Therefore a formal conditional logic without transitivity for conditionals is incomplete in the intuitive sense.

9.4 COROLLARY. The problem of conditionals cannot be solved in any formal logic.

9.5 REMARK. There are many who have tried to construct formal logical systems intended to solve the problem of conditionals. It is an empirical fact that nobody has ever succeeded in giving a satisfactory solution. Proposition 9.2 and Corollary 9.4 explain why nobody has and nobody ever will succeed.

2-10 Foundations of Logic

10.1 Informal Logic. Proposition 9.2 shows that if the problem of conditionals is solvable at all, it can be solved only in an informal logical theory. Conditionals have a central place in logic. Logic is concerned with inferences. A conditional \( A \rightarrow B \) justifies an inference from \( A \) to \( B \). This suggests that logic should be informal rather than formal. We are thus led to question formalisation as a universally valid fundamental principle in the foundations of logic.

10.2 Semantics of Formal Logic. The symbols of a formal system have no meaning in themselves. The logical constants of a formal logic must be given a meaning in a semantics. To say that a logical constant like, e.g., \( \rightarrow \) has meaning such-and-such is to make a statement in the metalanguage of the formal language. E.g., in the usual semantics of classical sentential logic, the meanings of the connectives are given by the truth conditions like (4-1)-(4-4). They belong to the metalanguage. For a formal language and logic there is no way to avoid this step to the metalanguage.

Suppose we have a formal logic \( L \) with an intended conditional operator \( \rightarrow \). \( \rightarrow \) can be a successful conditional operator only if a formal sentence like \( A \rightarrow B \) can in the semantics of \( L \) be interpreted as a conditional. But since the semantics of \( L \) is formulated in the metalanguage, the interpretation of \( A \rightarrow B \) is a conditional in the metalanguage. An example is Equivalence (4-4). The interpreting conditional belongs to the metalanguage. Therefore it talks about elements of the object language and is based on a functional connection at the object language level. An example is given by Proposition 7.2. There the conditional on the right-
hand side of the equivalence is based on a truth-functional connection on the object language level. We have the following observation.

10.3 OBSERVATION. Let \( A \rightarrow B \) be a conditional in any formal logic \( L \). Then \( A \rightarrow B \) must in the semantics of \( L \) be interpreted as a conditional based on a functional connection at the object language level. But since \( A \rightarrow B \) is a sentence in the object language, a correct interpretation of it must be a conditional based on a functional connection at the object level.

10.4 REMARK. A main task of logic is to give a theoretical analysis and representation of inferences. Most of the inferences we perform in practical contexts, in the sciences, and in mathematics are based on functional connections at the object level. It is a highly peculiar operation to let all inferences be based on functional connections at the object language level, one level higher. The natural approach would be to let such inferences be based on functional connections at the object level also in the theoretical model.

10.5 EXAMPLE. In Example 5.3, the conditional

\[
(10-1) \quad \begin{align*}
\text{IF you insert 5 kroner and you press button B,} \\
\text{THEN you will get a mug of coffee with milk and without sugar}
\end{align*}
\]

is based on a functional connection in the automation, thus at the object level. It is not based on the truth-functional connection at the object language level:

\[
(10-2) \quad \begin{align*}
\text{IF you insert 5 kroner and you press button B' is true,} \\
\text{THEN 'you will get a mug of coffee with milk and without sugar' is true}
\end{align*}
\]

10.6 EXAMPLE. In Example 5.4, the conditional

\[
(10-3) \quad \begin{align*}
\text{If } x = 2, \text{ then } y = 4
\end{align*}
\]

is based on the function in \( \mathbb{R}^2 \), and thus at the object level,

\[
y = f(x) = x^2
\]

In Example 5.5, the conditionals

\[
(10-4) \quad \begin{align*}
\text{If } x = 1, \text{ then } y = 0
\end{align*}
\]

\[
(10-5) \quad \begin{align*}
\text{If } x = 0, \text{ then } y = +1 \text{ or } y = -1
\end{align*}
\]

are based on the following functional connection in \( \mathbb{R}^3 \), also at the object level,

\[
\begin{align*}
&\begin{cases}
  z = x^2 + y^2 \\
  z = 1
\end{cases}
\end{align*}
\]

10.7 OBSERVATION. A reality based conditional

\[
(10-6) \quad \begin{align*}
\text{If } A \text{ then } B
\end{align*}
\]

implies the truth-function

\[
(10-7) \quad \begin{align*}
\text{If 'A' is true then 'B' is true}
\end{align*}
\]

But the inference from (10-7) to (10-6) need not be valid.

10.8 EXAMPLE. In the logical consequence (6-1) in Example 6.2, the conditionals \((O \rightarrow D)\) and \((T \rightarrow E)\) in the premises are true both as reality based conditionals and as information based truth-functions. The conditionals \((O \rightarrow E)\) and \((T \rightarrow D)\) in the conclusion are both false as reality based conditionals while as truth-functions at least one of them is true. This explains how the informal statement in Example 6.2 can be false while its formalisation (6-1) is true. Example 6.2 illustrates nicely a negative consequence of the unavoidable interpretation of formal conditionals as based on functional connections, in this case truth-functional connections, at the object language level.

10.9 A Programme. The preceding considerations give rise to a vision of an informal logic for conditionals based on functional connections at the object level (or whatever level one is talking about). To give the meaning of a conditional

\[\text{If } A \text{ then } B\]

we should need to define the (type of) functional connection on which the conditional is based. Thus an informal logic will be a theory of functional connections and their relations to conditionals and other informal sentences. Logic changes from the format of a formal calculus to the format of an informal theory.
Such a logical theory can presumably be based on concepts and results on functions from set theory, category theory, automata theory, and system theory. We have at our disposal mathematical instruments for such a logical theory which were not available to Frege when he created formal logic more than hundred years ago. We are not quite as restricted in our possibilities as he was.

10.10 ANALYSIS. Here is an attempt to analyse a conditional between two sentences A and B when A and B belong to the same language level.

If A then B =df there is an automaton (or other functional connection) A in some state S such that A is a description of an input event for A and B is a description of the corresponding output event.

10.11 The T-Principle. The T-principle
(10-8) 'A' is true iff A contains two conditionals. For each conditional, the antecedent and the consequent are on different language levels. Therefore Analysis 10.10 cannot be applied directly to the T-principle. We consider the direction (10-9) If A, then 'A' is true.

It is not clear how conditional (10-9) should be analysed. I sketch two possibilities.

10.12 Concept Formation in the T-Principle. The conditional (10-9) seems not to be based on a mechanical process as in an automaton. Rather it demands an intelligent conceptual process in the mind.
(1) The fact that A must be comprehended and represented by a proposition or sentence 'A'.
(2) By a step of reflection, 'A' must be compared with the fact A and found to agree with it. This conceptual operation may or may not be performed. Therefore the T-principle is not based on a functional connection. With reference to the figure in § 8.2, we see that the T-principle in this interpretation cannot be used to establish a functional connection between A and B. It only establishes a move from A being the case to B being the case.

10.13 The T-Principle is Information Based. There is a way to have the T-principle based on a functional connection if we see the T-principle as information based. Equivalence (10-8) is then really the following two conditionals:

(10-10) If 'A' is true, then 'A' is true is true
(10-11) If 'A' is true, then 'A' is true is true

Let L = L α be the given object language and let L β be its metalanguage, L γ its metametalanguage, etc. Let

\[ L_\alpha = \cup_\beta L_\beta \]

For X ∈ L, define the raising operator R and the lowering operator L:

<table>
<thead>
<tr>
<th>X</th>
<th>R(X)</th>
<th>L(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>TT</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>FP</td>
<td>F</td>
</tr>
</tbody>
</table>

Conditional (10-10) is based on the raising operation R. Conditional (10-11) is based on the lowering operation L.

Conditionals (10-10) and (10-11) are thus based on functional connections; but they cannot be used, as in the proof of Proposition 7.8, to establish a functional connection between A and B since A and B are not used, only mentioned in (10-10) and (10-11).

10.14 Transitivity. The Sylogism Principle is the logical consequence
(10-12) A → B, B → C ⇒ A → C
sometimes called transitivity of →. If A → B and B → C are reality based on the automata A and B, respectively, then A → C is reality based on the connected automaton (A B). If one of A → B or B → C is information based, then A → C is information based only. In an informal logic with explicit reference to the functional connections on which conditionals are based, there is no difficulty in handling transitivity of conditionals.

10.15 Ontology. In the philosophy of formal classical logic, it is common to draw a sharp distinction between logic and ontology. The logic is supposed to apply equally well to all kinds of reality. This is a conse-
sequence of the fact that the meanings of the logical operators are defined by functional connections at the object language level. Since this level is distinct from the object level with which the ontology is concerned, it is possible to uphold the sharp distinction.

For an informal logic based on functional connections at the object level, the situation may be different. Which logical laws are valid depends on which functions are supplied by the ontology and which not. Hansen (1989, 1996) give reasons to believe that the logic of quantum ontology is different from the logic of the ontology of macrophysics.

10.16 Simplicity. Classical formal predicate logic has the quality of being very simple. To define the meanings of the logical constants, we need study only a handful of simple functions. E.g., the logical constant $\rightarrow$ is defined by the simple functional connection studied in the proof of Proposition 7.2.

An informal logic as suggested in Programme 10.9 is probably more complicated. The number of different types of functions which must be studied is larger, possibly indefinite. Complications also arise from the ontology dependence of informal logic. Is it really worth the price to replace the simple formal logic with a complicated informal logic?

Note first that formal logic will not be lost. It is easy to recover it from the foundations of informal logic. First define information based logical constants, as in Proposition 7.2. Then apply the $T$-principle, as in Proposition 7.8. The result is ordinary formal logic. Second, Example 6.2, as well as examples in Section 11, show that there are problems which cannot be solved in any formal logic. Therefore we do not lose anything by revising the foundations of logic according to Programme 10.9. Possibly we will win something.

10.17 Applications of Formal Logic. In § 1.3, I have listed some of the beneficial consequences of formalisation in logic. None of these consequences are lost in the programme of § 10.9. Formalisation is exercised from the foundations of logic but not from logic. In § 10.16, it was indicated how formal logic can be recovered from the foundations of informal logic. Therefore none of the beneficial consequences of formal logic will be lost.

10.18 Non-Fregean Logic. Fregean logic is the usual classical formal logic. The informal logic suggested in § 10.9 may properly be called non-Fregean logic.

Frege initiated and stressed the importance of a strict adherence to the distinction between use and mention. This is a special case of, and can be seen as an anticipation of, the distinction between object language and metalanguage. In the analysis in Section 2-8, we saw that formal logic is a result of a systematic violation of this distinction by the use of the $T$-principle. Our diagnosis of what is wrong about Fregean logic is that in one respect this logic is not sufficiently Fregean.

2-11 Importance of the Problem

11.1 Introduction. The problem mentioned in the head-line is the inability of a formal conditional like material implication to represent completely informal conditionals. Example 6.2 is simple, striking, and apt for analysis. In this section, we consider other examples of greater theoretical or practical importance.

11.2 Dispositions. A disposition is a proneness to a certain reaction. Examples of dispositions are

$$(11-1) \quad \text{obedience, elasticity, fragility, generosity, inventiveness.}$$

11.3 Logical Representation. Maybe the most serious negative consequence of the paradoxes of implication is the impossibility of giving natural representations of dispositions in formal predicate logic. In natural languages, dispositions are represented by conditionals. Thus

$$(11-2) \quad x \text{ is dissolvable in water}$$

is represented by

$$(11-3) \quad \text{For every time } t, \text{ if } x \text{ is immersed in water at } t, \text{ then } x \text{ dissolves at } t$$

But this conditional cannot be represented by material implication $\rightarrow$. 

38
(11-4) \( x \) is dissolvable in water iff \( \forall t (x \text{ is immersed in water at } t \rightarrow x \text{ dissolves at } t) \)

The reason is that equivalence (11-4) implies that any object which never gets in touch with water is dissolvable.

Here we can see that definition (11-4) becomes wrong because \( \rightarrow \) is based on a functional connection at the object language level and not on a functional connection at the object level. The dissolubility of \( x \) is a consequence of the physical and chemical properties of the molecules in \( x \). These properties determine a functional connection in \( x \) itself (and thus at the object level) which for the input event \( x \text{ is immersed in water} \) produces the output event \( x \text{ dissolves} \). Therefore we must represent the disposition in the logic by a functional connection also at the object level. As shown, this can be done only in an informal logic.

11.4 Quantum Mechanics. The EPR-Bell experiment is concerned with the spin values of entangled pairs \((p_A, p_B)\) of spin-1/2 particles. We consider the spin values of three directions \( a, b, c \) which are known to lead to contradiction with quantum mechanics (QM). For a given particle \( p \), we use the abbreviations:

- \( x\)\( + \): \( x \) has spin up (spin \( +1 \)) in direction \( x \)
- \( x\)\( - \): \( x \) has spin down (spin \( -1 \)) in direction \( x \)
- \( x(p) \): \( p \) is measured in direction \( x \)
- \( x(\neg p) \): the result of measuring \( p \) in direction \( x \) is \( +1 \)
- \( x(\neg p) \): the result of measuring \( p \) in direction \( x \) is \( -1 \)

The sentence \( x\)\( + \) expresses a disposition in \( p \) to elicit a certain instrument reaction when \( p \) is measured for spin in direction \( x \):

(11-5) \( x\)\( + \Leftrightarrow (x(p) \Rightarrow x(\neg p)) \)

We see that conditional logic is involved in the analysis of the EPR-Bell experiment.

The analysis in Hansen (1996) has given the following results on the conditional logic (QRL) of a local realistic quantum theory:

(I) The conditional operator in (11-5) cannot be interpreted as material implication. It must be interpreted as an informal connective based on the quantum ontology.

(II) The basic foundational principle of classical logic that every sentence is either true or false cannot be valid for QRL. In QRL, some sentences may be without any truth-value.

(III) In the EPR-Bell experiment, exactly two of the sentences \( a\)\( + \), \( b\)\( + \), \( c\)\( + \) have truth-values for \( p_A \) and for \( p_B \) when \( a, b, c \) are pairwise incompatible. It is indeterminate until the moments of measurement which two of \( a\)\( + \), \( b\)\( + \), \( c\)\( + \) have truth-values and which has not.

(IV) The rules of inference

\[
\begin{align*}
A & \rightarrow B \\
A & \land B \\
A & \\
\forall x & A(x)
\end{align*}
\]

from classical logic cannot be valid in QRL, without restrictions.

11.5 Linguistics, Psychology, Computer Science. A task of linguistics is to give an analysis and theoretical model of natural languages. Examples 6.2 and 11.3 show that formal logic is not adequate for this task. An informal logic as the one suggested above may be a better tool for linguistic theory.

A task of cognitive psychology is to analyze and model the processes in the mind when we make logical inferences. As noted above, most of our inferences are based on the knowledge of functional connections at the object level rather than on truth-functional connections at the object language level. Therefore informal logic probably provides a better model of cognitive processes than formal logic.

In computer science, the logic used in artificial intelligent (AI) systems is formal logic. Thus, e.g., the programming language Prolog is based on Horn clause logic, a fragment of formal predicate logic. The human mind is still enormously superior to any AI system in processing conceptual information. The human mind normally uses informal logic, not formal logic. This suggests that the exclusive use of formal logic in AI may be a cul-de-sac. The way forward for more intelligent systems may go via informal logic.

Next we turn to Tarski's theorem on the concept of truth in formalized languages.
11.6 DEFINITION. Let $T$ be a theory in $L(T) \supseteq L(PA)$ where $L(PA)$ is the language of Peano arithmetic $PA$. A truth definition in $T$ is a formula $W(x)$ of $L(T)$ having only $x$ free such that for every sentence $A$ of $L(T)$,
\begin{equation}
\vdash_T A \leftrightarrow W(\#A^*)
\end{equation}
where $\#A^*$ is the Gödel number of $A$.

11.7 LEMMA (Fixed-Point Lemma). Let the formula $C(x)$ of the extension $T$ of PA have only $x$ free. Then there is a sentence $B_0$ in $L(T)$ such that
\begin{equation}
\vdash_T B_0 \leftrightarrow C(\#B_0^*)
\end{equation}
PROOF:
There is a recursive function
\[ \text{sub}: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \]
such that for any formula $A(x)$ and any term $t$ of $L(T)$,
\begin{equation}
\text{sub}(\#A(x)^*, \#t^*) = \#A(t)^*
\end{equation}
Since sub is recursive, it is representable in $T$. We can therefore assume that sub is available in $T$ (or rather in an expansion by definitions of $T$).

Let $C(x)$ be given. Define
\begin{equation}
B_0 = C(\text{sub}(\#C(\text{sub}(x,x))^*, \#C(\text{sub}(x,x))^*))
\end{equation}
By (11-8) and (11-9),
\begin{equation}
\text{sub}(\#C(\text{sub}(x,x))^*, \#C(\text{sub}(x,x))^*)^* = \#C(\text{sub}(\#C(\text{sub}(x,x))^*, \#C(\text{sub}(x,x))^*))^*, \#C(\text{sub}(x,x)^*)^* = \#B_0^*
\end{equation}
Since the defining axiom for sub in the expansion by definitions of $T$ is an equivalence, we get from (11-9) and (11-10)
\begin{equation}
B_0 \leftrightarrow C(\#B_0^*)
\end{equation}

11.8 REMARK. Note that what the proof of the lemma shows may be rephrased:
For any Gödel numbering of $L(T)$ and any formula $C(x)$ in $L(T)$ there is a number $\alpha \in \mathbb{N}$ such that
\[ \#C(\alpha)^* = \alpha \]
This number $\alpha$ is the fixed-point.

11.9 THEOREM (Tarski). Let $T$ be a consistent extension of PA. Then there is no truth definition in $T$.
PROOF:
Suppose there is a truth definition $W(x)$ in $T$. Consider the formula
\begin{equation}
\vdash_T \neg W(x).
\end{equation}
By the lemma, there is a sentence $B_0$ in $T$ such that
\begin{equation}
\vdash_T B_0 \leftrightarrow \neg W(\#B_0^*)
\end{equation}
By (11-6),
\begin{equation}
\vdash_T \neg B_0 \leftrightarrow W(\#B_0^*)
\end{equation}
From (11-12) and (11-13) follows immediately that $T$ is inconsistent:
\begin{equation}
\vdash_T W(\#B_0^*) \leftrightarrow \neg W(\#B_0^*)
\end{equation}
which contradicts the hypothesis of the theorem.

11.10 INTERPRETATION. The usual interpretation of Tarski's theorem is that $T \supseteq PA$ cannot contain no definition of truth; but this is too weak. The theorem implies that $T$ cannot contain no truth predicate, whether defined or primitive.

The theorem is only concerned with formal arithmetic languages and theories. It is often interpreted, even by Tarski himself, to apply also to informal languages having the means for syntactic reference, including self-reference. This is satisfied by most natural languages, including English. Since English contains a truth predicate, it seems to follow that English is inconsistent.

The informal proof of the inconsistency of English is like the proof of Tarski's theorem. Equivalence (11-6) takes the following form:
For all English sentences $A$,
\begin{equation}
A \leftrightarrow \#A^* \text{ is true}
\end{equation}
This is the T-principle. Since English contains the possibility of self-reference, we can form the sentence
\begin{equation}
\text{This sentence is false}
\end{equation}
This is the well-known Liar Paradox. Call the sentence (11-16) for $B_0$. Then
\begin{equation}
B_0 \leftrightarrow \#B_0^* \text{ is false}
\end{equation}
The equivalences (11-15) and (11-17) lead to a contradiction as in the proof of the theorem.
The conclusion that natural languages like Danish and English are inconsistent is hard to accept since they work so well in practice.

11.11 ANALYSIS. The proofs of Lemmas 11.7 and Theorem 11.9 are logically correct. The sentence (11-12) can indeed be obtained in formalised arithmetic with an underlying formal logic. But it is not trivial that Tarski's theorem can be extended to apply to informal languages and to informal theories as in Interpretation 11.10. Above we have found that there are essential differences and deviations between formal and informal logic. It has not been excluded that these differences are of importance for the possibility of having a truth predicate in an informal language.

Consider again the sentence

(11-16) This sentence is false

The sentence belongs both to the object language and to its own metalanguage. Thus the sentence (11-16) is only possible if we blur the distinction between object language and metalanguage. Such a blurring is allowed and exploited systematically in formal logics as we have seen above. An informal logic must adhere to a much sharper, possibly an absolute, separation between object language and metalanguage. A declarative, informal sentence must express a statement. Note that it is not easy to see what is stated by sentence (11-16).

As long as we do not have a well-developed and precise informal logic, it is not possible to give an exact treatment of the problem of truth predicates in informal logics and languages.

2-12 Other Solutions

12.1 Introduction. There have been many attempts before to solve the problem of conditionals. According to Kneale and Kneale (1962), it was debated lively already in antiquity. Characteristic of modern attempts are:

(1) They consist in devising formal systems which are claimed to be more representative than classical logic.

(2) They have all failed.

Proposition 9.2 implies that (2) is an inevitable consequence of (1).

We might be content with this general conclusion; but it may be instructive to analyse some of these attempts.

12.2 Christensen and Nasraklit. N. E. Christensen (1968, 1965, 1973) and L. Nasraklit (1987) have devoted much time and energy to the problem of conditionals. Christensen tested several formal systems without being satisfied with any of them. Proposition 9.2 and Corollary 9.4 show why this is inevitable. Nasraklit studies a sequence of truth-tables based on the ideas of truth-value gaps and many-valued logic. The semantical becomes increasingly more complicated to cope with ever new difficulties showing up. Proposition 9.2 shows that the sequence will never terminate in a satisfactory theory.

12.3 Assertibility. E. Adams, R.P. Grice, and F. Jackson base their "theories" of conditionals on the idea of assertibility. Adams denies that conditionals have truth-conditions; they only have assertibility conditions. Grice and Jackson distinguish between the truth-conditions and the assertibility conditions of conditionals. The truth-conditions of "If A then B" are given by the truth-functional conditional. The assertibility conditions are in Grice's case determined by roles for conversation and in Jackson's case by the conditional probability P(B/A). (See Read (1994) or Jackson (1987)). Example 6.2 and Remark 6.4 show that these accounts are inadequate.

12.4 The Ramsey Test. The following remarks in Ramsey (1929, 1978) are known as the Ramsey test:

"In general we can say with Mill that 'If p then q' means that q is inferable from p, of course, from p together with certain facts and laws not stated but in some way indicated by the context. This means p ⊃ q follows from these facts and laws." — "If two people are arguing 'If p will q?' and are both in doubt as to p, they are adding p hypothetically to their stock of knowledge and arguing on that basis about q."

There has been some closely related attempts to work out Ramsey's proposal like Stalnaker (1968), Stalnaker (1970), the former modified by
agreement with this world and not because of their agreement with a relation between possible worlds.

12.8 Entailment. We may make the following distinction:
(1) general conditionals;
(2) (logically) necessary conditionals, called entailment;
(3) logical consequence.

General conditionals and entailments are operations. Logical consequence is a relation. Entailments are special cases of conditionals. The equivalence

\[(12-1) \quad \vdash A \rightarrow B \quad \text{ iff } \quad A \models B\]

implies that there is a close relation between entailment and logical consequence.

The problem of entailment is the dissatisfaction with entailments like

\[(12-2) \quad \vdash A \wedge \neg A \rightarrow B\]

i.e., a logically false sentence entails anything, and

\[(12-3) \quad \vdash A \rightarrow (B \vee \neg B)\]

i.e., a logically true sentence is entailed by anything. They may be considered special cases of the well-known "paradoxes"

\[(12-4) \quad \neg A \vdash A \rightarrow B\]

\[(12-5) \quad B \vdash A \rightarrow B\]

Some authors have developed theories of entailment rather than of general conditionals. Examples are W. Ackermann, and A. Anderson and N. Belnap. (See Anderson and Belnap (1975-1992).)

12.9 An Argument. The following argument is sometimes ascribed to Albert of Saxony, sometimes to Pseudo-Scotus. We start with the following seemingly plausible logical principles:

\[(12-6) \quad A \wedge B \vdash A\]
A \land B \Rightarrow B
A \Rightarrow A \lor B
A \lor B, \neg A \Rightarrow B

We get the deduction:

(1) \quad A \land \neg A
(2) \quad A
(3) \quad \neg A
(4) \quad A \lor B
(5) \quad B

Premise:
from (1), by (12-6)
from (1), by (12-7)
from (2), by (12-8)
from (3), (4), by (12-9)

We have proved
A \land \neg A \Rightarrow B

and hence by (12-1),
\text{from } A \land \neg A \Rightarrow B

A system of entailment without the principle (12-2) must be without one of (12-6)- (12-9), or possibly the transitivity of logical consequence. The most influential system of entailment, the one by Anderson and Belnap, denies the validity of the principle of Syllogism (12-9). The following example shows that this is an unhappy choice.

12.10 EXAMPLE (Hanson (1982)). Note first that Principle (12-9) is valid in the semantics of classical formal logic. In the semantics, the premises express the following:

A \lor B: \text{at least one of } A, B \text{ is true}

\neg A: \text{ } A \text{ is false}

From this follows logically that B must be true. Hence B. It is of no relevance whether the principle can be used in a realistic context. The following parlour game nevertheless gives such an application.

I show you the box W and the two small objects a and b. I show you that there are the following four possibilities:
a is in W, b is in W;
a is in W, b is not in W;
a is not in W, b is in W;
a is not in W, b is not in W.

We use the parameters

A: a is in W
B: b is in W

I ask you to close your eyes, place b in W and a outside W. I give you the two pieces of information:

at least one of a and b is in W
a is not in W

I ask you:

What about b, is it in W or not?

You can infer:

b is in W

You have correctly used the pattern of inference

(12-9) \quad A \lor B, \neg A \Rightarrow B

It follows that Anderson and Belnap’s system of entailment is intuitively incomplete. This example gives an illustration of Proposition 9.2.

12.11 REMARK. Work on the problem of entailment can be seen as an attempt to understand better the nature of logical consequence and therefore the nature of logic. My belief is that such an understanding will come naturally when we have developed better foundations of logic along the guide-lines in sections 9 and 10.

2-13 Summary and Conclusion

13.1 The problem of conditionals has nothing to do with truth-value gaps, many-valued logics, modal logics, or belief revision. Instead it is a problem in the foundations of logic. It is an inevitable consequence of the use of formalisation. It can only be solved in an informal logical theory.

13.2 In the semantics of any formal logic, all inferences are based on functional connections at the object language level. This is unnatural since the inferences which are modelled normally are based on func-
tional connections at the object level. In some respects, the principle of formalisation is highly artificial. Informal logic seems to be a more natural approach.

13.3 Frege established the foundations of formal logic. This was the starting point of more than hundred years of impressive development of symbolic and mathematical logic. Now time may have come for a revision and development of the foundations of logic along the guide-lines sketched in this paper. This may lead to a revitalisation of logic and may also vitalise the foundations of other disciplines directly depending on logic: mathematics, probability theory, quantum mechanics, and philosophy.

Notes

*) Parts of the content have been presented at lectures and seminars at Uppsala Katedralskola 1993 and at the universities of Uppsala, Umeå, and Tartu 1994. Two or three paragraphs are attempts to answer objections and questions from Thorild Dahlquist and Sten Lindström. In a much earlier phase, I had stimulating discussions on conditionals with Lennart Nørreklit and the late Niels Egmont Christensen. It was through a lecture by Christensen that I first came to know about the problem of conditionals.

Section 7. The proofs of propositions 7.2 and 7.8 are from Hansen (1986). The present article may be seen as an attempt to analyse the paradox in that article.

References


Christensen, N. E. (1973), “Is there a “Logic” or Formal System Based on the Concept of Truth Determinant?” Danish Yearbook of Philosophy, 10, 77-85.


3. What is Philosophy?

3-1 Introduction

1.1 PROBLEM. We want to know what philosophy is, or at least to find a partial answer to that question.

1.2 Importance of the Problem. Here follow some reasons why and how an answer to the question of the nature of philosophy may be useful.

1. An answer gives guide lines for posing new problems in philosophy.

2. An answer functions as foundations for philosophy. It gives stability and solidaity to thinking. For a socially inclined person, who gets his philosophical problems from discussions and the meanings of his philosophical activities from the reactions of others, this is of little importance. But for the lonely outsider, who follows his own way and his own destiny, it is of importance.

3. An answer helps to develop the methodology of philosophy.

4. An answer helps to develop our conception of philosophy and to develop philosophy itself. A wrong or too narrow or too broad definition is better than no definition at all. Once an answer has been attempted, it is easier to formulate an alternative answer. Otherwise the question is left to silence, obscurity, and passivity. They are enemies of all theoretical progress.

1.3 IDEA. The opinions in the history of philosophy on the nature of philosophy show considerable divergences. It seems to be generally agreed, however, that a philosopher is a thinker and that philosophy is a theoretical activity.
The basic idea in the present paper is that since philosophy is theoretical, a study of the logical structure of theories may throw light on the nature of philosophy. In particular, we will be interested in the structure of such theories which are important in theoretical methodology: number theory, the theories of the fields of real and complex numbers, set theory, and category theory. Thus this paper is an investigation into the logical foundations of philosophy.

3.2 Arithmetical Theories

2.1 General Structure of Theories. A theory $T$ is determined by

(I) a language $L(T)$,
(II) a logic,
(III) a set of nonlogical axioms.

We take a theory $T$ to be any set of formulas of $L(T)$ such that $T$ is closed under the logic. Thus $T$ is the closure under logic of its set of nonlogical axioms.

2.2 Language and Logic. The language $L(T)$ will here be assumed to be a first-order predicate logical language. Higher order languages will not be considered. One reason for this is that first-order languages and logic seem to have the expressive and deductive power needed for all mathematical and scientific theories. This is Hilbert's thesis. Another reason is that the mathematical logic of first-order theories is well developed and comparatively simple while the mathematical logic of higher order languages is still quite rudimentary and much more difficult.

The logic will be assumed to be classical formal first-order predicate logic. The challenge of intuitionistic logic and the observations in Chapter 2 of the present book show that there are still open problems in the foundations of logic; but for the moment these problems will be disregarded.

2.3 EXAMPLE. (I) The language of group theory $G$ is $L(G) = \{ *, e \}$

where $*$ is a binary function symbol and $e$ is a constant.

(II) The language $L(PA)$ of Robinson arithmetic $Q$ and Peano arithmetic $PA$ is

$L(PA) = \{ 0, S, +, \cdot, < \}$

where $0$ is a constant, $S$ is a unary function symbol, $+$ and $\cdot$ are binary function symbols, and $<$ is a binary predicate.

2.4 EXAMPLE. Very many axiomatisations of predicate logic are known. They all consist of

(I) a possibly empty set of logical axioms,
(II) a nonempty set of rules of inference.

We do not decide for one axiomatisation rather than another. We simply assume that in any theory $T$ we have all logical truths of the language $L(T)$, all logically valid consequences in $L(T)$, and all valid rules of inference in $L(T)$.

2.5 Nonlogical Axioms. The set of nonlogical axioms of $T$ is any set of sentences of $L(T)$. The set may be empty or nonempty, finite or infinite, recursive or nonrecursive.

2.6 EXAMPLE. (I) Many axiomatisations of group theory $G$ are known. An example is:

G1. $\forall x \forall y \forall z x*(y*z) = (x*y)*z$  
G2. $\forall x x*e = x$  
G3. $\forall x \exists y x*y = e$

(II) $G$ can be expanded in many ways. The theory of commutative groups $CG$ is obtained by adding to the set of axioms of $G$ the further axiom:

G4. $\forall x \forall y x*y = y*x$

$G$ and $CG$ are finitely axiomatised theories.

2.7 EXAMPLE. (I) Robinson arithmetic $Q$ has the following axioms:

Q1. $\forall x S(x) \neq 0$  
Q2. $\forall x \forall y (S(x) = S(y) \rightarrow x = y)$  
Q3. $\forall x x + 0 = x$  
Q4. $\forall x \forall y x + S(y) = S(x + y)$
2.10 Models. Let $G_0$ be the theory in $L(G)$ with no no logical axioms. Then $G_0$ is the predicate logic for $L(G)$. The class $M(G_0)$ of models for $G_0$ is the class of all models for $L(G)$. For any theory $T$ in $L(G)$,

$$M(T) \subseteq M(G_0).$$

Let $A_0$ be the theory in $L(PA)$ with no no logical axioms. Then $A_0$ is the predicate logic for $L(PA)$. The class $M(A_0)$ is the class of all models for $L(PA)$. Again we have for any theory $T$ in $L(PA)$,

$$M(T) \subseteq M(A_0).$$

Any theory in $L(G)$ can be obtained from $G_0$ by adding new axioms. Any arithmetical theory can be obtained from $A_0$ by adding new axioms. Any true arithmetical theory can be obtained from $A_0$ by adding new axioms whose all true in $T$.

2.11 Constraining Axioms of G. We start with $G_0$ and the class $M(G_0)$ of all models for $L(G)$. For some models $M$ in $M(G_0)$, the axiom $G_1$ is true:

$$M \models G_1$$

For other models $M$ in $M(G_0)$, $G_1$ is false:

$$M \models \neg G_1$$

By adding $G_1$ to $G_0$, we constrain the set of models of $G$ by excluding all models satisfying (2-2). Further constraints on $M(G)$ are imposed by adding $G_2$ and $G_3$.

Consider the set of sentences in $L(G)$:

$$S = \{G_1, \neg G_1, G_2, \neg G_2, G_3, \neg G_3, G_4, \neg G_4\}$$

Any subset of $S$ can be used as a set of constraints for a theory in $L(G)$. Of course, some subsets of $S$ give rise to inconsistent theories.

From the point of view of pure mathematics, the only condition on a set of axioms selected from $S$ is that it is consistent. From the point of view of applied mathematics, further conditions are imposed. A selection of axioms must be appropriate for the class of applications which the constructor of the theory has in mind. This means that he intends a certain subset of $M(G_0)$. Historically, group theory was defined by Galois to express structural properties common for a class of number systems including $\mathbb{Z}, \mathbb{Q}, \mathbb{Q}^+, \mathbb{R}, \mathbb{R}^+, \mathbb{C}$. When defining $G$, we must then see to it that $\mathbb{Z}, \mathbb{Q}, \mathbb{Q}^+, \mathbb{R}, \mathbb{R}^+, \mathbb{C} \in M(G)$. 

Q5. \(\forall x \cdot 0 = 0\)
Q6. \(\forall x \forall y \cdot S(y) = x \cdot y + x\)
Q7. \(\forall x \cdot x < 0\)
Q8. \(\forall x \forall y \cdot (x < S(y) \land x < y \land x = y)\)
Q9. \(\forall x \forall y \cdot (x < y \land x + y = y \land y < x)\)

(II) An axiomatisation of PA (with some redundancies) can be obtained from Q by adding PA.1-PA.9 coincide with Q.1-Q.9 and adding an axiom scheme

PA.10 \(A(0) \land \forall x \cdot (A(x) \rightarrow A(S(x))) \rightarrow \forall x \cdot A(x)\)

where $A(x)$ is any formula of $L(PA)$. PA.10 are the induction axioms.

(III) Complete arithmetic $Th(\mathfrak{N})$ is obtained by letting its language be $L(Th(\mathfrak{N})) = L(PA)$ and its set of nonlogical axioms be the set of all sentences of $L(PA)$ which are true in the standard model $\mathfrak{N}$ of arithmetic.

Q is finitely axiomatised. PA is recursively but not finitely axiomatisable. $Th(\mathfrak{N})$ is not recursively axiomatisable.

2.8 DEFINITION. (I) An arithmetical theory is any theory in $L(PA)$. Thus an arithmetical theory is obtained by taking as nonlogical axioms any set of sentences in $L(PA)$. The class $A$ of arithmetical theories is the set

$$\mathcal{A} = \{ [T] \mid T is a theory in L(PA) \}$$

(II) A true arithmetical theory is any theory in $\mathcal{A}$ having $\mathfrak{N}$ as a model. The class $\mathcal{T}$ of true arithmetical theories is the set

$$\mathcal{T} = \{ [T] \mid T is a theory in L(PA) \}$$

(\forall Q, PA_0, and Th(\mathfrak{N}) all belong to $\mathcal{T}$)

2.9 Classification of Axioms. In the next paragraphs, I give a classification of the nonlogical axioms of arithmetical theories. True arithmetical theories are, of course, of special interest. For illustration, even group theory $G$ will be considered because of its simplicity. The nonlogical axioms of true arithmetical theories fall in three classes:

(1) constraining axioms;
(2) defining axioms;
(3) axioms of insight.

This classification will now be explained.
2.12 Construction and Analysis. Let L be any language, $T_0$ the theory in L with no neotological axioms, and T any theory in L obtained from $T_0$ by adding new constraining axioms.

T may be intended as a pure construction. As to verification, the only thing to verify is that T is consistent. For many theories, consistency proofs are possible; but Gödel's incompleteness theorem imposes severe restrictions on the possibility of giving consistency proofs for the more interesting and powerful theories.

T may be a construction intended for certain applications. The group theories G and CG belong to this category. It should still, if possible, be verified that T is consistent; but there is now also a problem of verifying that T is appropriate for the intended applications.

Finally, T may be intended as a reconstruction of an existing and informal concept already in use or as a representation of an existing phenomenon. Peano arithmetic belongs to this category. Apart from the consistency of T, there is the problem of verifying that T gives a true reconstruction of the concept or a satisfactory representation of the phenomenon. Note that these reconstructions can be seen as special cases of the constructions for certain applications considered above. A reconstructing theory is a construction intended to be applicable to a given concept or a given set of phenomena. Note also that it was shown in Hansen (1995), Chapter 2 of the present volume, that some concepts cannot be adequately reconstructed in any formal system.

An analysis is a reconstruction of an existing informal concept or existing phenomenon. Therefore analyses are just special constructions where one strives to make the construction resemble something existing. An analogy with the art of painting may be useful. A nonfigurative painting is a pure construction. A figurative painting is a construction where the painter tries to reconstruct, make the picture resemble, something in reality. In both cases, the painting is a construction of oil-paint on canvas. We see that the opposition between analyses and constructions is only apparent.

The working out of theories which are pure constructions or constructions intended for certain applications by using constraining axioms is a task undertaken in philosophical construction. The working out of theories intended to reconstruct a given concept by using constraining axioms is a task undertaken in philosophical analysis. We see that the opposition between construction and analysis in philosophy is only apparent. Analysis is a special form of construction.

The practitioners of philosophical constructions and philosophical analysis claim that they are involved in a scientific activity. Our analysis confirms this. The need and possibility of consistency proofs and of verification or falsification of the appropriateness of the construction or of the similarity with the concept analysed shows that these enterprises are indeed scientific to some degree.

2.13 Constraining Axioms of $Q$ and PA. We start with $A_0$, the predicate logic of $L(PA) = \{ 0, S, +, \cdot, < \}$. Like with $Q_0$, there is a large number of ways of adding constraining axioms to $A_0$, thereby defining the properties of 0, S, +, \cdot, and <. Up to a certain point in the development of a theory, we have great freedom in choosing constraining axioms. This point is reached when we have a theory sufficiently rich to allow the proof of Gödel's theorem. As with theories in $L(G)$, we may look at the theories in $L(PA)$ as pure constructions limited only by the demand of consistency, as constructions intended for certain applications, or as analyses of given concepts or phenomena. In the case of number theories, we actually have just one model in mind: the standard model $\mathbb{N}$. The development of an arithmetical theory is therefore a reconstruction; and we are limited to true arithmetical theories, i.e., members of $\mathcal{G}$. The important thing about constraining arithmetical axioms is that they define the properties of the individual 0, of the operations S, +, \cdot, and of the relation < by excluding certain models where 0, S, +, \cdot, < have other properties.

Gödel's theorem can be proved for Robinson arithmetic $Q$. As far as is known, it cannot be proved for any theory T having as its set of axioms a proper subset of $\{ Q_1, \ldots, Q_9 \}$. Suppose the arithmetical theory is developed by successively adding $Q_1, \ldots, Q_9$ to $A_0$. Then at any stage up to the addition of $Q_9$ we have freedom to add the next $Q_i$ or add $\neg Q_i$. I claim that when all of $Q_1, \ldots, Q_9$ are added and Gödel's theorems are provable, then the further development of arithmetic takes another character. This idea will be explored later in the treatment of axioms of insight.

2.14 Defining Axioms. The second kind of axioms in § 2.9 are defining axioms. Suppose we have a theory T in a language L(T). We want to de-
fine a new symbol $U$ not occurring in $L(T)$. Form a new language $L(T^*) = L(T) \cup \{U\}$. Then form the theory $T^*$ by adding a new neologistical axiom to $T$ which defines $U$. How this is done will now be explained.

2.15 Definition of a Predicate. Let $x_1, \ldots, x_n$ be distinct variables and $B(x_1, \ldots, x_n)$ be a formula of $L(T)$ having no other free variables than $x_1, \ldots, x_n$. Choose a new $n$-ary predicate $P$ not occurring in $L(T)$. Let $L(T^*) = L(T) \cup \{P\}$. Let $T^*$ be $T$ extended by the neologistical axiom

$$P(x_1, \ldots, x_n) \leftrightarrow B(x_1, \ldots, x_n)$$

Let $A^*$ be a formula of $L(T^*)$. Using the defining axiom (2-3) and substitution of equivalents, $A^*$ can be transformed to a formula $A$ of $L(T)$ such that

$$T^* \vdash A^* \iff T \vdash A$$

2.16 Definition of a Function Symbol. Let $x_1, \ldots, x_n, y, z$ be distinct variables and $B(x_1, \ldots, x_n, y)$ a formula having no other free variables than $x_1, \ldots, x_n, y$ and no bound occurrence of $z$. Assume the following can be proved in $T$:

$$T \vdash \forall x_1 \ldots \forall x_n \exists y B(x_1, \ldots, x_n, y)$$

$$T \vdash \forall x_1 \ldots \forall x_n \forall y \forall z (B(x_1, \ldots, x_n, y) \land B(x_1, \ldots, x_n, z) \rightarrow y = z)$$

(2-5) is the existence condition; (2-6) is the uniqueness condition. Let $f$ be a new $n$-ary function symbol not occurring in $L(T)$ and $L(T^*) = L(T) \cup \{f\}$. Form $T^*$ from $T$ by adding the defining axiom of $f$:

$$y = f(x_1, \ldots, x_n) \leftrightarrow B(x_1, \ldots, x_n, y)$$

Let $A^*$ be a formula of $L(T^*)$ containing $f$. Using Axiom (2-7) and substitution of equivalents, $A^*$ can be transformed to a formula $A$ of $L(T)$ such that

$$T^* \vdash A^* \iff T \vdash A$$

If we in this procedure let $n = 0$, we get a method for defining a new constant $c$.

2.18 EXAMPLE. In group theory $G$, Axiom G3 gives the existence condition

$$G \vdash \forall x \exists y x*y = e$$

The following results are easily proved in $G$ in the given order

(2-10) $G \vdash \forall x \forall y (x*y = e \rightarrow y*x = e)$

(2-11) $G \vdash \forall x \exists y c = x*y$

(2-12) $G \vdash \forall x \forall y \forall z (x*y = c \land x*z = c \rightarrow y = z)$

(2-12) is the uniqueness condition. Let $L(G^*) = L(G) \cup \{t\}$ where $t$ is a unary function symbol. Then $t$ can be defined by the defining axiom

$$T \vdash \forall x \forall y (y = x' \leftrightarrow x*y = e)$$

2.19 REMARKS. (I) The defined formula $P(x_1, \ldots, x_n)$ may be seen as an abbreviation of the defining formula $B(x_1, \ldots, x_n)$. The defined formula $y = f(x_1, \ldots, x_n)$ is an abbreviation of the defining formula $B(x_1, \ldots, x_n, y)$. Similarly, the formula $A^*$ of $L(T^*)$ in equivalences (2-4) and (2-8) is an abbreviation of the formula $A$ of $L(T)$.

(II) Defining axioms have a function very different from that of constraining axioms. With constraining axioms, we stick all the time to the same language $L(T)$. Let $T$ be given and $M(T)$ its class of models. Let $A$ be a new constraining axiom such that $T \models A$. Then the effect of adding $A$ to $T$ is to exclude some models from $M(T)$, namely all those models where $\neg A$ is true.

With a defining axiom, we move to a new extended language $L(T^*)$, but not one single model in $M(T)$ is excluded. Let $M^* = (M, \ldots, U)$ be a model of $T$. Let $U$ be the new defined symbol. Then the defining axiom for $U$ gives a method of expanding $M$ to a unique model $M^* = (M, \ldots, U)$ of $T^*$. This mapping gives a bijection of $M(T)$ and $M(T^*)$.

2.20 Construction and Analysis. Just as with constraining axioms, defining axioms may be conceived as pure constructions, as constructions intended for certain purposes, or as reconstructions analyses.

Definitions of new terms and concepts and analyses in the form of definitions of terms and concepts already existing in language is traditionally seen as a task for philosophy.

Our analysis of the logical struct-
ture of theories has confirmed that there is indeed room for such a task in the development of a theory.

The founding fathers of analytic philosophy (Frege, Russell, Moore) seem to have considered definitions as in §§ 2.15-2.16 as the only task of philosophical analysis. Our examination of the structure of theories shows that this is too narrow a conception of analysis. To handle all analytic and constructive problems, we need both the method of defining axioms and the method of constraining axioms. The later development in analytic philosophy is in agreement with this insight.

Constructivist and analytic philosophers use to boast of their philosophy as being scientific. If one follows the methodology of defining axioms in §§ 2.15-2.16, the addition of a defining axiom to a consistent theory T can never lead to an inconsistent theory T'. This follows from the fact that T' provably is a conservative extension of T. Consistency proofs are therefore irrelevant in this context. But as with constraining axioms, there are verifiable or falsifiable statements on the degree to which an extension by definitions is suitable for intended applications. There are verifiable or falsifiable statements on how well a defining axiom gives a true analysis of a given concept. In these respects, the method of defining axioms is susceptible to scientific control.

2.21 Axioms of Insight. We assume that we work in the language L(PA) and that the development of the true arithmetic has reached an axiomatised level $T_0$ where Gödel's theorem can be proved. $T_0$ may, e.g., be Q or PA. What is demanded of $T_0$ is that all recursive relations and functions are representable in $T_0$. Since $T_0$ is incomplete, there is room for further development of $T_0$. New axioms will exclude some models from $S(T_0)$, but it turns out that we have much less freedom to choose new axioms as we have with constraining axioms. The methods of metamathematics generate in a transfinite process a sequence of new true axioms. The proof of Gödel's theorem defines a process which, if followed, leads to an arithmetic insight expressed in the Gödel sentence $Con_{T_0}$. Turing's theorem implies that if this kind of insight is only repeated sufficiently many times an essentially more complete number theory results.

Now follows an analysis of axioms of insight in arithmetic. We need some theorems from mathematical logic.

2.22 DEFINITION. Let $T$ be a theory in L(PA). $T$ is Ω-consistent iff there is no formula $A(x)$ in L(PA) such that

$$T \models A(0), \quad T \models A(1), \ldots, \quad T \models \exists x \neg A(x)$$

2.23 THEOREM (Gödel's First Incompleteness Theorem). Assume that $T$ is a theory in L(PA) such that $T$ is axiomatisable and $Q \subseteq T$. Then there is a sentence $\Phi$ which, interpreted in the metalinguage, asserts its own unprovability such that

(1) if $T$ is consistent, then $T \not\models \Phi$;
(2) if $T$ is Ω-consistent, then $T \not\models \neg \Phi$.

2.24 REMARK. J. B. Rosser showed that statements (1) and (2) in Gödel's first theorem can be strengthened to

(1') if $T$ is consistent, then $T \not\models \Psi$;
(2') if $T$ is consistent, then $T \not\models \neg \Psi$,

where $\Psi$ is a slightly more complicated sentence than $\Phi$.

2.25 THEOREM (Gödel's Second Incompleteness Theorem). Let $T$ be a theory in L(PA) such that $T$ is axiomatisable and $Q \subseteq T$. Then

$$T \models Con_{\text{Const}_T}$$

where $Con_{\text{Const}_T}$ is the sentence of L(PA) asserting the consistency of $T$.

PROOF:

Nice expositions of the proofs of Gödel's and Rosser's theorems can be found in Smoryński (1977).

2.26 DEFINITION. Let $T$ be any theory with a recursive language L(T). Let a fixed Gödel numbering for L(T) be given.

(1) Pr$_T$ is the proof relation for $T$ satisfying

$$Pr_T(m, n) \iff m \text{ is the Gödel number of a proof in } T \text{ and } n \text{ is the Gödel number of the last formula in this proof.}$$

(2) Thry$_T$ is the theorem predicate for $T$ satisfying

$$Th_{\text{ry}}(n) \iff n \text{ is the Gödel number of a theorem of } T \iff \exists m \text{ Pr}_T(m, n)$$
2.27 THEOREM. Let $T$ be any theory with a recursive language $L(T)$. Let a fixed Gödel numbering of $L(T)$ be given. Then
(I) $T$ is axiomatisable $\iff$ $\text{Th}_T$ is recursively enumerable.
(II) $T$ is axiomatisable $\iff$ $\text{Pr}_T$ is recursive.

PROOF:
Simple proofs of (I) occur in Shoenfield (1967) and in Bell and Machover (1977).
(II) $\Rightarrow$: If $T$ is axiomatisable, then there is a recursive axiomatisation of $T$. Then it is decidable whether a given sequence of formulas in $T$ is a proof and which formula is the last in the sequence. Therefore $\text{Pr}_T$ is decidable and hence, by Church's Thesis, recursive.
(II) $\Leftarrow$: If $\text{Pr}_T$ is decidable, $\text{Th}_T$ is RE by Definition 2.26(II). By (I) of the present theorem, $T$ is axiomatisable.

2.28 EXAMPLE. We define a sequence $T_0, T_1, \ldots$ of theories in $L(PA)$. Let $T_0$ be $Q$ or $PA$. Define
$$T_1 = T_0 + \text{Cons}_T$$
where $\text{Cons}_T$ is the sentence which expresses that $T_0$ is consistent. Now $T_1$ is consistent and axiomatisable and by Gödel's second theorem
$$T_1 \models \text{Cons}_T$$

Let $T_2 = T_1 + \text{Cons}_T$. We thus get an infinite sequence $T_0, T_1, \ldots$.

Let
$$T_\xi = \cup_{\eta<\xi} T_{\eta}$$
Then $T_\xi$ has an RE set $\text{Ax}_\xi$ of axioms:
$$\text{Ax}_\xi = \text{Ax}_{PA} \cup \{\text{Cons}_T, \text{Cons}_T, \ldots\}$$
It follows that $\text{Th}_{T_\xi}$ is RE and therefore, by Theorem 2.27, $T_\xi$ is axiomatisable. We can therefore continue the process into the realm of transfinite ordinals and form
$$T_0, T_1, \ldots, T_\xi, T_{\xi+1}, \ldots, T_{\xi+\xi}, \ldots$$

2.29 DEFINITION. (I) Let $T$ be a true, axiomatisable, arithmetical theory in $L(PA)$ sufficiently powerful to allow the proof of Gödel's theorem. A reflection principle of $T$ is any sentence of the form
$$\text{Th}_T(\star A\star) \rightarrow A$$

where $A$ is any sentence of $L(PA)$ and $\star A\star$ is the Gödel number of $A$.
(II) Define
$$\text{RFN}_T = \{\text{Th}_T(\star A\star) \rightarrow A | A \text{ is a sentence in } L(PA)\}$$
i.e., $\text{RFN}_T$ is the set of reflection principles of $T$.

2.30 REMARK. The content of the sentences in $\text{RFN}_T$ is that any theorem of $T$ is true. This clearly implies that $T$ is consistent. On the other hand, $T$ may be consistent without being sound. It follows that $\text{RFN}_T$ is logically stronger than $\text{Cons}_T$.

2.31 EXAMPLE. A more powerful extension of a theory $T$ can be obtained by adding $\text{RFN}_T$ as new axioms instead of $\text{Cons}_T$. We define a new sequence by letting $T_0$ be $Q$ or $PA$. Let
$$T_1 = T_0 + \text{RFN}_T, \quad T_2 = T_1 + \text{RFN}_T, \ldots$$
$$T_\xi = \cup_{\eta<\xi} T_{\eta} = T_0 + \cup_{\eta<\xi} \text{RFN}_T$$
Since $(\text{RFN}_T \in \text{RE})$ is an RE set of recursive sets, $\cup_{\eta<\xi} \text{RFN}_T$ is RE. Therefore $T_\xi$ has an RE set of axioms and therefore is RE. By Theorem 2.27, $T_\xi$ is axiomatisable. We may therefore continue the process to get an ordinal sequence of theories
$$T_0, T_1, \ldots, T_\eta, T_{\eta+1}, \ldots, T_{\eta+\eta}, \ldots$$
This sequence will now be defined.

2.32 DEFINITION. We define an ordinal sequence $\{T_\xi | \xi \in \text{On}\}$ of theories in $L(PA)$:
$$T_0 = \text{any true theory for which Gödel's theorem can be proved}$$
$$T_{\beta+1} = \begin{cases} T_\beta + \text{RFN}_T & \text{if } T_\beta \text{ is axiomatisable} \\ T_\beta & \text{otherwise} \end{cases}$$
$$T_\xi = \cup_{\eta<\xi} T_{\eta} \quad \text{if } \xi \text{ is a limit ordinal}$$
$$T_\alpha = \cup_{\eta<\alpha} T_{\eta} \quad \text{if } \alpha \text{ is a limit ordinal}$$

2.33 DEFINITION. (I) A number theoretic theorem is a sentence of the form
$$F(x) = 0 \text{ for infinitely many } x \in \mathbb{N}$$
where $F$ is a primitive recursive function.

(II) A theory $T$ in $L(PA)$ is complete with respect to number theoretic theorems if every true number theoretic theorem can be proved in $T$.

2.34 THEOREM (Turing). Let $(T_n)$ be an ordinal sequence of theories obtained by adding reflection principles as in Definition 2.32. Then there is a countable ordinal $\gamma$ such that

(I) $T_\gamma = T_m$;

(II) $T_\gamma$ is not axiomatisable;

(III) for every $\beta<\gamma$, $T_\beta$ is axiomatisable and $T_\beta \subseteq T_\gamma$;

(IV) $T_\gamma$ is complete with respect to number theoretic theorems.

PROOF:

(I), (II), and (III) are trivial. (IV) follows from a result in Turing (1939); see Davis (1965), p. 195. Here it is important that $\gamma_{\text{pso}_{1}}$, the smallest nonrecursive ordinal.

2.35 REMARK. The definition 2.32 of an ordinal sequence of theories is somewhat different from Turing's. In particular, it is less constructive. Turing assigns theories only to recursive ordinals. Theorem 2.34 is modified accordingly.

2.36 OPEN PROBLEMS. (I) Is $T_\gamma = T_m$ complete? None of the theories, called "ordinal logics", defined by Turing is complete. But since his definitions are more constructive than mine, the incompleteness of his theories need not carry over to mine. My conjecture is that $T_\gamma$ is incomplete.

(II) Does it make any difference for the final result $T_\gamma$ whether we start with $T_0=Q$ or $T_0=PA$? In particular, if $T_0=Q$ will $PA\subseteq T_\gamma$?

2.37 REMARK. Though it is unknown whether $T_\gamma$ is complete. Theorem 2.34 shows that $T_\gamma$ is more complete than $Q$ and $PA$. Number theoretic theorems are important in connection with metamathematical problems. Are there any strictly metamathematical incompletenesses in $PA$ which are removed by adding reflection principles? This question has been answered affirmatively in Paris and Harrington (1977). There is a theorem of finitary combinatorics, provable in $\text{ZF}$, which can be expressed in $L(PA)$. It cannot be proved in $PA$; but it can be proved in $PA$ extended with all reflection principles of the form

$$\forall x \text{ Thm}_{PA}(\forall x A(x)) \rightarrow \forall x A(x)$$

where $A(x)$ has only $x$ free.

We see that the reflection principles contain new essential mathematical and metamathematical information.

2.38 Reflection Principles. We now try to analyse the nature of the insights incorporated in the reflection principles. We try to prove the following:

(2-13) $\forall \beta (T_\beta$ is true $\rightarrow T_{\beta+1}$ is true)

(2-14) $T_0$ is true $\rightarrow \forall \beta (T_\beta$ is true)

(2-15) $(\forall \beta<\gamma) (\text{RF}_{\beta}$ is true)

(2-15) follows from (2-14) by the definition of $T_{\gamma+1}$. (2-14) follows from (2-13) by ordinal induction.

Now consider (2-13). Assume $T_\beta$ true. Let $A$ be any sentence in $L(PA)$. How do we see that $(\text{Thm}_{PA}(\forall x A(x)) \rightarrow A)$ is true? First we make a jump from the object language $L(PA)$ to the metalanguage of $T_\beta$. Next we see that $T_\beta$ can be incorporated in $T_\beta$'s metalanguage and, via the Gödel numbering, be applied to $T_\beta$ considered as a syntactic structure. With this interpretation, it is clear that all $(\text{Thm}_{PA}(\forall x A(x)) \rightarrow A)$ must be true. For suppose some is false, e.g., $(\text{Thm}_{PA}(\forall x A(x)) \rightarrow B)$ is false. Then we have a true arithmetic $T_\beta + \neg(\text{Thm}_{PA}(\forall x A(x)) \rightarrow B)$ which is false in some concrete interpretation (application), namely a metamathematical interpretation. This is impossible.

I claim that this "proof" of (2-13) is not a strictly logical proof. The reason is that the jump from the object language to the metalanguage of $T_\beta$ combined with the insight that $T_\beta$ can be applied to $T_\beta$ itself is not a mechanical logical process. It is a creative operation which generates a new insight, namely that all $(\text{Thm}_{PA}(\forall x A(x)) \rightarrow A)$ are true. Let us call this combined operation the reflection operation because of its similarity with self-consciousness.

2.39 Construction and Analysis. The extension of $T_\beta$ to $T_{\beta+1}$ is not a pure construction; for they are limited only by the demand that $T_{\beta+1}$ should be consistent. The extension of $T_\beta$ is limited also by the demand that $T_{\beta+1}$ should be metamathematically true. We see that when the axiomatic development of arithmetic reached a point where Gödel's
Theorem is provable, then we no longer have the same freedom in the further development as before that point is reached.

\( T_{\beta+1} \) can be seen as a construction designed for certain applications, namely the metamathematical applications to \( T_{\beta} \). The important thing is that this construction is based on the insight gained by the reflection operation. Since Gödel's theorem is provable for \( T_{\beta+1} \), \( T_{\beta} \) has an application to \( T_{\beta} \) itself. This application gives an interpretation of \( T_{\beta} \) which is real and concrete and cannot be deleted by a constraining axiom. If we to \( T_{\beta} \) add the negation \(-\lnot(\text{Thm}_n(\alpha \rightarrow B))\) of some suitable reflection principle in \( \text{RFN} \beta \), we get a theory which is true in some proper subclass of the class of models of \( T_{\beta} \); but we also get a theory which is false in the concrete metamathematical interpretation of \( T_{\beta} \). The latter interpretation cannot be excluded from the class of interpretations of \( T_{\beta} \) as an abstract set theoretical model can.

The insight on which the axioms in \( \text{RFN} \beta \) are based are not the result of a conceptual analysis only. This is a consequence of the fact that (2.13) in § 2.38 cannot be proved by logic alone. The creative aspect of the reflection operation goes beyond analysis. Analysis never create anything. They only make explicit what is already implicitly there.

We conclude that reflection principles are another kind of axioms than constraining axioms and defining axioms. They are axioms based on insights reached by the reflection operation. Such insights reached by reflection have traditionally, at least in some periods, been considered a task for philosophy.

The development of axioms of insight by reflection operations is not a scientific enterprise. A theory \( T \) is scientific if it has a mechanical proof or verification relation. By Church's thesis and Theorem 2.37, such theories coincide with axiomatisable theories. The creative aspect of the reflection process makes it non-mechanical and therefore non-scientific. Note that this is compatible with the complete absence of arbitrariness. Anybody who goes through the reflection operation as suggested in § 2.38 will come to the same insight.

### 2.40 Classification of Axioms

It is possible to argue that the tripartite classification of axioms

1. constraining axioms,
2. defining axioms,
3. axioms of insight

is complete. This is done in the same way as in § 3.5 for the axioms of set theory.

#### 2.41 OPEN PROBLEMS

1. Normally, it is given as a necessary condition for a statement or theory \( S \) to be scientific that it can be tested intersubjectively whether a proposed proof or verification is correct. There is an ambiguity in the term "intersubjectivity". In the objective interpretation, the criterion demands that a machine or automation can do the check. This is the interpretation used in § 2.39. In the subjective interpretation, it suffices that any two persons who do the check will always reach the same result. It is not trivial that the two interpretations are equivalent. In the example with the reflection principles in \( \text{RFN} \beta \), the principles can be seen to be true by any two persons; but it seems that they cannot be verified by any machine.

We should not only know if the subjective and the objective interpretations are equivalent. If they are not, we need to know which interpretation is more adequate for the criterion of scientificity.

2. In § 2.39, the notion of a concrete interpretation of a theory was introduced. When arithmetic reaches a certain level of development, the theory can be applied to itself. This gives a concrete interpretation of the theory in addition to the abstract set theoretical models. Moreover, the concrete interpretation cannot be deleted by a constraining axiom as an abstract model can. This is why the concrete interpretation forces the Gödelian axioms like \( \text{Com}_3 \) and \( \text{RFN} \beta \) upon us.

We need a better understanding of the concrete interpretations and their relations to the usual models of a theory.

### 3.3 Other Theories

#### 3.1 Discrete Mathematics

Every theorem of discrete, finitary mathematics can be interpreted as a theorem of \( \Pi \). Therefore the conclusions on the axioms of arithmetic apply to discrete finitary mathematics and all problems where a discrete, finitary methodology suffices.
3.2 Mathematical Analysis. A more powerful methodology is given by the theories of the field \( \mathbb{R} \) of real numbers and the field \( \mathbb{C} \) of complex numbers. They are sufficient for most of physics.

3.3 Set Theory and Category Theory. Still more powerful theoretical methodologies come from set theories like ZF and ZFC and extensions of category theory like the theory CS of the category of sets and topos theory \( \mathcal{O} \). Together set theory and category theory contain all theoretical methodology known and used today. It is clearly of particular interest for our purposes to study their axiomatic structures. First I make a comment on unified methodology.

3.4 Unified Methodology. We consider the three foundational systems ZFC, CSC (the theory of the category of all sets with the axioms of choice), and topos theory \( \mathcal{O} \). CS and the topos theory \( \mathcal{O} \) are both extensions of category theory C. We first compare ZF and CS, each possibly extended with AC.

ZF gives a natural development of the set universe. But it is unsatisfactory that the set universe itself cannot be treated as an object in any set theory. Moreover, big classes like the category of all sets and the category of all groups, mappings between them called functors, categories of categories, categories of functors, and natural transformations from a functor to another all occur naturally in category theory. They cannot be defined in ZF. Category theory itself gives a framework for the solution of the problem. We can have big classes as objects in the domain if we treat them as black boxes. This is the strategy used in CS. Black boxes are in physics characterised by their causal relations to other systems. Black boxes are in mathematics characterised by their functional relations to other systems. Those functional relations are the functors between the categories.

ZF develops the set universe in a natural way; but it does not allow the reconstruction of all the mathematical structures used in category theory. This problem is solved in CS. Negative in CS is that the reconstruction of several set theoretic structures is cumbersome and somewhat unnatural. More important is that not all of set theory can be formulated in the language of CS. In particular, the axiom of replacement cannot be expressed in CS. This problem is solved in topos theory \( \mathcal{O} \) which provably is an extension of both ZFC and CSC. Topos theory is the most general foundational system for mathematics known. It gives a unification of all known theoretical methodology. But it is still hampered by the same unnaturalness in reconstructions as CS is.

For our purposes, it is important that all of classical mathematics can be reconstructed in any of ZFC, CSC, and \( \mathcal{O} \). In particular, Peano arithmetic and analysis can be reconstructed.

3.5 Classification of Axioms. We get the same tripartite classification of the axioms of set theory in the language \( L(ZF) = \{ e \} \) as for arithmetic. The presence of

1. constraining axioms,
2. defining axioms

is trivial. Since PA is implicit in ZF, Gödel's theorem can be proved for ZF. We can therefore define an ordinal sequence \( \{ T_0 \} \) of theories by letting \( T_0 = ZF \), or \( T_0 = \) a suitable fragment of ZF, and proceed as in Definition 2.32 by adding reflection principles. Now Turing's theorem 2.34 can be proved. We therefore have the third category of axioms:

3. axioms of insight (based on reflection).

Since it is not known whether the theory \( T_m \) obtained by this process is complete, there may be other types of axioms which can be seen to be true in other ways than by those already mentioned. Since they can be seen to be true, they are also axioms of insight. It may be that there are problems which can be formulated in \( L(ZF) \) but which are in principle unsolvable. Answers to such problems may be left open in set theory. Alternatively, any of the possible answers may be added to the set theory as a constraining axiom, a free construction limited only by the demand of consistency. Thus the tripartite classification of axioms

1. constraining axioms,
2. defining axioms,
3. axioms of insight

is complete also for set theories though we have not been able to give a complete map of the routes of insight.

The same analysis applies to CS and to topos theory. Since all of PA can be developed in these theories, we get the same three types of axioms.
3-4 Philosophy

4.1 Introduction. In this section, I try to draw some consequences for the understanding of the nature of philosophy from the analyses and results in sections 2 and 3. First we define the task of philosophy within the framework of formalised first-order theories.

4.2 First Definition. Within the framework of formalised first-order theories, the task of philosophy is to formulate and motivate the non-empirical axioms of a theory. The axioms fall in three categories:

(1) constraining axioms,
(2) defining axioms,
(3) axioms of insight.

The axioms may be either free constructions limited only by the demand of consistency, or constructions for certain applications, or reconstructions/analyses of informal concepts or theories, or sentences expressing insights generated by, e.g., reflection operations.

4.3 Remark. In some respects, this definition is too limited since it applies only to work within the framework of formalised first-order theories. There is no reason why philosophy cannot question the validity of this framework. Thus, in the article "Conditionals and the Foundations of Logic", Chapter 2 in the present volume, I challenge the validity of the foundations of formal logic. In some respects, therefore, the definition needs to be generalised.

4.4 Generalised Definition. The axioms of a theory belong to the foundations of the theory. So do the language and the logic of the theory. The foundations of a theory consists of all the fundamental principles of the theory construction. The superstructure of the theory are the theorems of the theory, the set of logical consequences of the foundations. A natural generalisation of Definition 4.2 which avoids the objection in Remark 4.3 is therefore:

The task of philosophy is to develop foundations.
Philosophical problems are foundational problems.
Philosophy is non-empirical foundational research.

The same characterisation can be applied to the development of foundations as to the development of the axiomatics of a theory: they are free constructions limited only by consistency, or constructions for certain applications, or reconstructions/analyses, or constructions based on insights, e.g., from the reflection process.

It is important to realise that not only mathematics and mathematical theories have foundations. The same is true of all theories and sciences, of the social sciences and humanitarian scholarship, of the arts and creation of art, of morals, of individual and collective decision making, of social organisation, national organisation and polity, and even of human life itself.

Next we consider three possible objections to the theory of philosophy in § 4.2 and the present paragraph.

4.5 Objection. The definition is too broad. Even others than professional philosophers, e.g., mathematicians, logicians, and physicists develop axioms and foundations.

Answer: This is correct. Most of the axiomatisation of mathematics and most of the work on its foundations has been done by mathematicians. Most of the development of the basic principles and foundations of physics has been done by physicists. But this does not contradict definitions 4.2 and 4.4. Philosophising, the work on philosophical problems, is by no means limited to professional philosophers and philosophy departments. Philosophising is one of the most widely spread theoretical activities (because it is so useful). We consider some examples. Bohr and Einstein were physicists, not professional philosophers. It does not prevent their work on the foundations of quantum mechanics from being philosophy. Richard Wagner was a composer, not a professional philosopher. He contributed to the development and explicit formulation in writing of the foundations of romantic music. This contribution certainly belongs to philosophy of music.

Developing the theory of philosophy in §§ 4.2 and 4.4, I have taken the following into consideration:
(1) distinctions occurring in reality, and
(2) what has occurred in the history of philosophy.

The distinctions between axioms and theorems, between foundations and superstructure are forced upon us by logic, and therefore they are
unavoidable components of reality. Demarcations between academic departments are (partly) social conventions. Work on foundational problems has, at least in some periods and by some philosophical schools, been considered a task for philosophy. It follows that as long as no distinction existing unavoidably in reality and as significant as the one between foundations and superstructure can be found which distinguishes between two kinds of nonempirical foundational problems, one kind philosophical and the other non-philosophical, then all non-empirical foundational problems are philosophical.

Since the theory of philosophy in definitions 4.2 and 4.4 is based on distinctions which exist in reality, it is a theory of the nature of philosophy, not just an analysis of the conventional use of the word 'philosophy' or of somebody's concept of philosophy.

4.6 OBJECTION. The scheme of definitions 4.2 and 4.4 is too narrow. Much philosophy goes beyond this scheme, e.g., Hegel's, Kierkegaard's, Nietzsche's, Heidegger's, Sartre's, and Wittgenstein's.

ANSWER:
To the extent that these philosophers develop or contribute to theories, they must fit into the scheme. For there are no other propositions in a theory than axioms and theorems. There are no other components of a theory than foundations and superstructure. Most of Kierkegaard's work, e.g., consists in developing the foundations of one possible way of life, the life of "a truly existing individual". People like Hegel, Nietzsche, and Heidegger do not fit into the scheme when they act as sages or prophets and make statements which they want to be taken as truths rather than as, e.g., foundations of one of several possible ways of doing things. That these activities do not fit into the scheme does credit to the scheme since, as sages or prophets, the named people do not act as philosophers.

The case of Wittgenstein is different. For him, philosophical problems were the results of bewilderment. The solution to a philosophical problem is not a theory. The problem disappears, and results in no theory at all, when the bewilderment is cleared up and removed. Wittgenstein can hardly have denied the existence and need of axioms and foundations. He may have denied that the development of them is a task for philosophy. But this is a quarrel about the use of the word 'philosophy'. The use I have made of it has at least as strong a support in the history of philosophy as Wittgenstein's use. We have seen that analytic work and thinking leading to insights are necessary in the development of the foundations of a theory. Traditionally such activities have been considered to be philosophic. The later Wittgenstein's purpose was only to criticise and leave reminders. In reality, he did not avoid making some positive theoretical statements. Some people even claim to see the rudiments of the foundations of a Wittgensteinian theory of language and language acquisition in his writings. But even Wittgenstein's purely critical purposes fit into my scheme of philosophy. Criticism of the foundations of theories in a field where theory is possible is a necessary condition for theoretical progress in the field. Wittgenstein's mistake was to believe that this is the only task of philosophy. To criticise the foundations of theories in a field where no theory is possible, as the late Wittgenstein claimed he did in his hermeneutical approach to language, is to do a theoretical cleaning work which is perfectly compatible with the definition of philosophy in §§ 4.2 and 4.4. Again his mistake is to claim that this is the whole task of philosophy.

4.7 OBJECTION. My article Hansen (1995), Chapter 2 in the present volume, shows that there are concepts which cannot be represented in any formal system. For some purposes, logic must be informal. The definition 4.2 of philosophy, generalised in Definition 4.4, is based on formal logic and properties of formal theories. When the foundations of an informal logic, as suggested in the article, is developed, the theory in definitions 4.2 and 4.4 may turn out to be completely wrong.

ANSWER:
The article on the foundations of logic shows indeed that formal logic and formal theories are inadequate for some purposes. This was one of the motives for the generalisation in definition 4.4 of the idea in Definition 4.2. The article also gave a proof which shows that there is a very close kinship between formal and informal logic. This close kinship makes it highly improbable that the theory of philosophy in §§ 4.2 and 4.4 should not persist after a revision of the foundations of logic, needing at worst only minor modifications.

4.8 DEFINITION. Philosophical analysis is a method. A philosophical analysis is a reconstruction and a making explicit of an existing informal concept, theory, language use, or text using formal predicate logic as theoretical framework. When keeping exclusively to a first-order lan-
Analytic philosophy has been accused of being too passive and retrospective by its exclusive concentration on the analysis of what is already there. I agree in this criticism. This is remedied in constructivist philosophy. Analytic philosophy has been accused of leaving aside the deeper and more important problems of philosophy. Again I agree. Analytical and constructions are indispensable parts of philosophy; but as shown in Section 2, the deepest problems of philosophy demand insights which go beyond analysis and construction. Analytic philosophy and constructivist philosophy are too narrow. Only the concept of philosophy in Definition 4.2 and 4.4 is sufficiently broad to do justice to the role philosophy must play in history.

4.11 Scientific Philosophy. For many philosophers, the ideal has been that philosophy should be a science. Carnap is an example. Much of analytic and constructivist philosophy satisfies this demand. It can be tested intersubjectively to what extent an analysis agrees with what is analysed. At least some constructed theories can be proved consistent or be proved to fit the applications for which it was constructed.

When we leave the realm of constructivist and analytic philosophy, the picture changes. Axioms based on insights cannot always be verified. The combinatorial principle in Paris and Harrington (1977), which is independent of PA, can be proved in ZF; but this is an exception. We consider the reflection principles in RNF\textsubscript{T} added to a true arithmetic theory T. In most cases, they cannot be proved in T. There is no existing entity against which they can be verified. Instead, by doing the reflection operation, we create a situation in which the sentences in RNF\textsubscript{T} can be seen to be true.

A necessary condition for scientificity in a mathematical theory T is the presence of a recursive proof relation Pr\textsubscript{T}. By Church's thesis, this is equivalent with the existence of the possibility of intersubjectively testing whether a proposed proof of a sentence in T really is a correct proof. By Theorem 2.27, this coincides with T being axiomatizable. When we go beyond T by adding a new axiom based solely on insight, we also transcend the proof relation Pr\textsubscript{T}. The only evidence for the axiom is the insight. Another person can reach the same insight by going through the same process which leads to it, but there is no mechanical method which yields the insight.
A strong motive for those who insist on a scientific philosophy is the conviction that the alternative is arbitrariness in philosophy. As horrifying examples, philosophers like Hegel, Nietzsche, Heidegger, Sartre, and Levinas in the tradition of German idealistic, hermeneutical, and phenomenological philosophy are mentioned. It should be pointed out that the development of new axioms based on insight is possible without leaving room for the slightest arbitrariness. In the development in §§ 2.32-2.38 of $T_{n+1} = T_n + \text{RFN}_n$, it is only possible to add the members of $\text{RFN}_n$ to $T_n$. If we add the negation of one single reflection principle in $\text{RFN}_n$ to $T_n$,

$$T' = T_n + \neg(\text{Thm}(\neg B) \rightarrow B)$$

we get a theory which can be seen by everybody to be false.

4.12 Exact Philosophy. Another objection to philosophy based on insights is that if we open for insights in philosophy, then we also open for loose talking and inexactness. The same examples may be offered as in § 4.11.

The example studied in §§ 2.32-2.38 showed that axioms based on insight are compatible with mathematical exactness. Indeed, it is only by exactness that it is possible to reach the insight in that example. The present writer is convinced that solid and lasting progress in philosophy can only be achieved by applying the methods and results of logic and mathematics and by striving, as far as possible, towards their standards of rigour.

4.13 Philosophy Education. In most philosophy departments, at least those where analytic philosophy is dominant, a course in logic uses to be an early ingredient in the curricula. The logic taught is almost always only formal logic in the object language, Frege's, Peano's, and Russell's logic. This logic is sufficient for analytic philosophy which uses only constraining and defining axioms. But it is not enough for constructivist philosophy which also occasionally needs to be able to give consistency proofs. And it is certainly not sufficient for the development of axioms based on insight. In the latter case, we need information about the theories which can only be reached in the metalanguage. Such information continues mathematical logic. The axioms of insight studied in §§ 2.32-2.38 are based on the reflection operation. This operation is studied in proof theory, recursion theory, and set theory.

The education of philosophers qualified to work with the deeper and more difficult problems of philosophy must include a thorough training in formal and mathematical logic. A philosopher without training in logic is as worthless as a physicist without training in mathematics. Logic is the philosopher's single most powerful theoretical tool.

4.14 The Pythagorean Conception of Philosophy. A tradition in antiquity dating back to the Plato pupil Heraclides Ponticus says that Pythagoras was the first to use the word philosophia. A sophos, sage, was a person who claimed to have knowledge. In the style of a prophet, the sage claimed that his beliefs were the truth and that he knew the truth. Pythagoras claimed only to be a philosophos, somebody who strives for knowledge. The usual interpretation of this is that Pythagoras thereby expressed the basic attitude of openness characteristic of good philosophy. He claimed only that his beliefs were hypotheses which were open to critical scrutiny by himself or other philosophers and which after critical scrutiny might turn out to be wrong. They could therefore not be labelled knowledge. His work was only a striving for knowledge.

There is another possible interpretation of what Pythagoras's thought may have been. Note first that in many cases, we can actually claim that we have knowledge. This is, e.g., the case when we have an algorithm to decide a question. Pythagoras's idea must therefore have been that in the development of theories (beliefs), there is a phase where we cannot claim to have knowledge. This phase is the task of philosophy.

As an example we consider the development of an axiomatic mathematical theory $T$. There are only two phases in the development of $T$:
(1) the development of the foundations, i.e., the axioms of $T$, and
(2) the development of the superstructure of $T$.

The development of the superstructure of $T$ consists exclusively in proving theorems of $T$. In this phase, we can claim to have knowledge. Since $T$ is axiomatisable, by Theorem 2.27, $T$ has a recursive proof relation $\text{Pr}_T$. Therefore once a proof $(B_1, \ldots, B_n)$ of a sentence $B_n$ is proposed, another mathematician, or a machine, can effectively check whether it is a correct proof in $T$ or not. If the proof turns out to be cor-
rect, then we know that $T \vdash B_0$. The Pythagorean form of philosophy must therefore be exercised in the development of the axioms of $T$. As we saw in Section 2, there is for the development of some types of axioms no recursive proof or verification relation. Notably for the axioms of insight based on the reflexion operation, no such proof is normally possible. In the case of these axioms, we can therefore not claim that we know them in the strict sense, only that they are expressions of a striving for knowledge.

The conception of philosophy defined in §§ 4.2 and 4.4 can therefore be seen as being in agreement with the original Pythagorean conception.

4.15 CONSEQUENCES. We consider a few consequences of Definition 4.2, 4.4 of the nature of philosophy.

(I) Philosophy is not a science. Though some parts of philosophy, notably those based on analysis, can be subjected to scientific control, this is not true of all philosophy. One task of philosophy is to develop and expand the foundations of science. Sometimes this demands that we must go beyond all existing proof and verification relations. Doing this, we also transcend the limits of scientificity. As shown above, this need not open for arbitrariness in philosophy.

(II) Philosophy is not the science of the most general features of reality. Philosophy is not concerned with more general features than the special sciences and academic disciplines. The foundations of a science has exactly the same degree of generality as the superstructure. For instance, the set of axioms of a mathematical theory is logically equivalent to its set of theorems. They therefore have the same content of information and the same degree of generality.

(III) Philosophy of physics is not only philosophy about physics but also philosophy in physics. Philosophy of morals is not only philosophy about morals but also philosophy in morals. To develop the foundations of a field is to do work in the field itself, expressed in the object language of the field.

(IV) Definition 4.2, 4.4 is based on the sharp distinction between the foundations of a science and the superstructure of the science. The philosophical task is to develop the foundations. The scientific task is to develop the superstructure. But there is no sharp distinction between what work is more natural to do for the professional philosopher and for the professional scientist. In some situations, it is natural and necessary for a physicist to turn his attention to the foundations of physics to develop his science. A philosopher who has contributed an innovation to the foundations of physics must draw at least some consequences belonging to the superstructure in order to check the soundness of his idea.

(V) There is no philosophical knowledge, i.e., there is no body of knowledge which belongs to philosophy. Philosophy is a process by which new insights are reached. When, e.g., an axiom is accepted, it is incorporated in the set of mathematical knowledge and not in philosophy. How is philosophical education possible if there is no philosophical knowledge? A philosophical education must consist in training in processes which lead to new foundational insights:

1. Knowledge of the logical foundations of philosophy as outlined in the present essay. This implies a thorough training in formal and mathematical logic.

2. Training in problem solving: logical, mathematical, and philosophical.

3. Some knowledge about previous philosophical work, i.e., about the history of philosophy. One semester of training in the history of philosophy easily suffices and more may be harmful.

4. A solid knowledge about the field(s) to whose foundations the philosopher intends to contribute.

4.16 The Meaning of Philosophy. The identification of philosophy with foundational studies makes philosophy a great and worthwhile enterprise. Philosophy is at the foundations of much of conscious and organised human activity. Whenever the development of the superstructure of one of the basic sciences like logic, mathematics, probability theory, or physics stops, slows down, or idles, then a revision and development of the foundations is almost always a necessary condition for a new start. The same is true for many other walks of life.
4. Logical Rationalism: A Programme

4.1 Introduction

1.1 History. In the 17th and 18th centuries, rationalism and empiricism were the two dominant philosophical movements. The leading philosophers of rationalism were Hobbes, Descartes, Spinoza, and Leibniz. The leading empiricists were Locke, Berkeley, and Hume. Kant, being essentially a rationalist, tried in his transcendental philosophy to combine the best elements of the two traditions. After Kant, rationalism deteriorated further and in practice disappeared as a force on the philosophical scene. With Kant's successors, the connection with reality became slender in much so-called continental philosophy, from Hegel and Nietzsche to Heidegger and Sartre. Arbitrariness replaced the stern intellectual discipline of a Descartes or a Spinoza.

Empiricism continued to flourish. At the beginning of the 20th century, empiricism was amalgamated with the new symbolic logic to form logical empiricism, the most powerful and influential form of empiricism ever. Though logical empiricism is no longer as vital as it was half a century ago, many of its ideas and results still live and exercise influence in contemporary analytic philosophy.

1.2 Purpose. The purpose of the present paper is to try to initiate a revival of rationalism. I will make an attempt to show that rationalism is based on a sound idea. I will try to show that a strong form of rationalism can be created by amalgamation of some ideas from classical rationalism with mathematical logic.
1.3 The Cogito. The basic principle of Descartes's epistemology is the

\[ \text{Cogito \ ergo \ sum.} \]

He used this as the starting point for an axiomatic epistemology, a project which soon went away.

My idea is that what is sound in rationalism is contained in the Cogito. 'Cogito' is usually translated as 'I think'. 'Cogito' actually has the broader meaning of pursuing something in the mind. I will take 'cogito' in the broad meaning of I am conscious or there is consciousness (in the universe). The question then is

\[ \text{(1-2)} \]

What follows from the mere fact that there is consciousness?

I will call this question the Cartesian question. Descartes draws the conclusion that the system of thoughts/consciousness exists. This system is then identified with 'I', giving the conclusion 'sum'. The next problem for Descartes is to prove the existence of a distinction between 'I' and the rest of the universe, and here things begin to go wrong for him. In the following, I will be interested only in the Cartesian question.

1.4 REMARK. The remarks in § 1.3 are not, and are not meant to be, a faithful exegesis of Descartes. An exact agreement with Descartes is not essential for the argument in the present article. What is important is the Cartesian question. I think this question, though not explicitly formulated, is implicit in Descartes's writings.

4.2 Preliminary Definitions

2.1 Language. We start with a natural, informal language like English. We then take a fragment \( F \) of English containing a set of names, a set of function terms, and a set of extensional predicates. We assume that the fragment is reasonably rich. Then we assume that the basic language of arithmetic and set theory is included in the fragment. We also assume that some empirical facts can be expressed in \( F \). Each expression in \( F \) has an associated standard interpretation determined by the meanings of the expression.

Next we define a formal first-order language \( L \) corresponding to this fragment of English. For every name in \( F \), we include a constant in \( L \). For every \( n \)-ary function term in \( F \), we include an \( n \)-ary function symbol in \( L \). For every \( n \)-ary predicate in \( F \), we include an \( n \)-ary predicate in \( L \). Thus there is a bijection \( \Phi \) between \( L \) and the set of names, function terms, and predicates in \( F \). This bijection \( \Phi \) is easily extended to a bijection between the informal sentences in \( F \) and the formal sentences in \( L \).

2.2 DEFINITION. (I) A sentence \( B \) in \( F \) is analytically true iff \( B \) is true and \( B \) can be shown to be true on the basis only of the meanings of the linguistic components of \( B \).

(II) A sentence \( B \) in \( F \) is synthetically true iff \( B \) is true and \( B \) is not analytically true.

(*The truth of \( B \) is defined relative to the standard interpretation.*)

2.3 DEFINITION. A sentence \( B \) of \( L \) is analytically (synthetically) true iff the associated sentence \( B^* \) in \( F \) under the bijection \( \Phi \) is analytically (synthetically) true.

2.4 DEFINITION. (I) A sentence \( B \) in \( F \) is an apriori truth iff \( B \) is true and \( B \) can be shown to be true without the use of empirical information.

(II) A sentence \( B \) in \( F \) is an aposteriori truth iff \( B \) is true and \( B \) can be shown to be true only by the application of some piece of empirical information.

2.5 DEFINITION. A sentence \( B \) of \( L \) is an apriori (aposteriori) truth iff the associated sentence \( B^* \) in \( F \) under the bijection \( \Phi \) is an apriori (aposteriori) truth.

2.6 REMARK. The four predicates defined in §§ 2.2-2.5 can be combined. Among the true sentences of \( L \) or \( F \), we get the following subcategories:

analytic apriori,
analytic aposteriori,
synthetic apriori,
synthetic aposteriori.
There is general agreement that there are sentences which are analytic apriori and sentences which are synthetic aposteriori. It is also clear that there are no sentences which are analytic aposteriori. As soon as B can be shown to be true only by analysing the meaning of the linguistic components of B, no support from empirical facts is needed. The disagreement is on the question whether there are sentences which are synthetic apriori truths.

2.7 DEFINITION. (I) Empiricism is the view that there are no synthetic apriori truths. In other words, in empiricism the set of analytic truths coincides with the set of apriori truths. The set of synthetic truths coincides with the set of aposteriori truths.

II) Rationalism is the view that there are synthetic apriori truths. In other words, in rationalism the set of analytic truths is a proper subset of the set of apriori truths. (* See Figure 4.1. *)

```
<table>
<thead>
<tr>
<th></th>
<th>apriori</th>
<th>aposteriori</th>
</tr>
</thead>
<tbody>
<tr>
<td>analytic</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>synthetic</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
```

Rationalism

```
<table>
<thead>
<tr>
<th></th>
<th>apriori</th>
<th>aposteriori</th>
</tr>
</thead>
<tbody>
<tr>
<td>analytic</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>synthetic</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
```

Empiricism

Figure 4-1. The two diagrams illustrate the difference between rationalism and empiricism. A box representing an empty class is shaded. A nonempty box is marked by a cross.

2.8 DEFINITION. (I) Let \( \mathfrak{A} \) denote the theory in \( L \) whose nonlogical axioms are the set of all analytically true sentences in \( L \). We call \( \mathfrak{A} \) the theory of analytic truths in \( L \).

II) Let \( \mathfrak{P} \) denote the theory in \( L \) whose nonlogical axioms are the set of all apriori true sentences in \( L \). We call \( \mathfrak{P} \) the theory of apriori truths in \( L \).

III) Let \( \mathfrak{T} \) denote the theory in \( L \) whose nonlogical axioms are the set of all sentences in \( L \) which are true in the actual universe \( U \). We call \( \mathfrak{T} \) the theory of truths in \( L \).

2.9 OBSERVATION. (I) \( \mathfrak{A} \) is a proper subtheory of \( \mathfrak{T} \), \( \mathfrak{A} \subseteq \mathfrak{T} \).

II) \( \mathfrak{T} \) is a complete theory in \( L \), i.e., for every sentence \( A \) in \( L \), either \( A \in \mathfrak{T} \) or \( \neg A \in \mathfrak{T} \).

2.10 REMARK. Observation 2.9 is correct only if \( \mathfrak{A} \) is a consistent theory. We will assume that \( \mathfrak{F} \) and \( L \) have been so chosen that this is the case. Thus, by Tarski’s theorem, we must, e.g., assume that \( \mathfrak{F} \) and \( L \) contain no truth predicate.

2.11 DEFINITION. If \( \Gamma \) is a set of formulas in \( L \), then we let \( M(\Gamma) \) denote the class of all models of \( \Gamma \).

2.12 REMARK. (I) We allow that \( M(\Gamma) \), apart from set theoretic structures, also contains concrete systems in which the closures of all formulas in \( \Gamma \) are true. E.g., the actual universe \( U \) may be in \( M(\Gamma) \).

II) Since \( \mathfrak{T} \) is complete, all models in \( M(\mathfrak{T}) \) are elementarily equivalent. For our purposes, we may assume that \( U \) is the only element of \( M(\mathfrak{T}) \), \( M(\mathfrak{T}) = \{ U \} \).

### 4-3 The Synthetic Apriori

3.1 PROBLEM. In this section, I make an attempt to show that there may be a way to define the apriori which implies the rationalist principle that there are synthetic apriori truths. I will use concepts from model theory.

3.2 ANALYSIS. We clearly have

\( (3-1) \quad \mathfrak{A} \subseteq \mathfrak{P} \subseteq \mathfrak{T} \)
This implies

\[(3-2) \quad (\mathcal{U}) \subset M(\mathcal{U}) \subset M(\omega)\]

For rationalism, we need

\[(3-3) \quad \mathcal{U} \subset \mathcal{\varphi} \subset \mathcal{U}\]

\[(3-4) \quad (\mathcal{U}) \subset M(\mathcal{\varphi}) \subset M(\omega)\]

Thus the problem for rationalism is to define a suitable class \(M(\mathcal{\varphi})\) of models which is strictly intermediate between \(U\) and the class \(M(\omega)\).

3.3 DEFINITION. A system is called self-referential if some truth about the system can be represented as knowledge in the system itself.

3.4 EXAMPLES. (I) The universe is a self-referential system. Through the cognitive activity of human beings, truths about the universe are represented as knowledge in our central nervous systems, in writings, or on magnetic tapes and thus in the universe itself.

(II) A human being is a self-referential system. Through self-consciousness, each of us knows much about himself.

(III) Let \(A\) be a sentence in the language \(L(\text{PA})\) of Peano arithmetic such that

\[(3-5) \quad \text{PA} \vdash A\]

Thus it is true about the system \(PA\) that \(A\) is a theorem of \(PA\). Let \(\text{Pr}_{\text{PA}}\) be the binary proof relation of \(PA\):

\[\text{Pr}_{\text{PA}}(x, y) \iff x \text{ is the } \text{Gödel number of a proof in } \text{PA and } y \text{ is the } \text{Gödel number of the last formula in the proof.}\]

By (3-5), there is a proof of \(A\) in \(PA\). Let \(p\) be the Gödel number of the proof and \(*A* \text{ the } \text{Gödel number of } A\). Then by (3-5) and the definition of \(\text{Pr}_{\text{PA}}\), \(\text{Pr}_{\text{PA}}(p, *A*)\). Since \(\text{Pr}_{\text{PA}}\) is recursive, it is representable in \(PA\) so that

\[(3-6) \quad \text{PA} \vdash \text{Pr}_{\text{PA}}(p, *A*)\]

Hence in turn,

\[(3-7) \quad \text{PA} \vdash \exists x \text{ Pr}_{\text{PA}}(x, *A*)\]

\[(3-8) \quad \text{PA} \vdash \text{Thm}_{\text{PA}}(*A*)\]

where \(\text{Thm}_{\text{PA}}\) is the unsolvability theorem predicate of \(\text{PA}\) which satisfies

\[\text{Thm}_{\text{PA}}(x) \iff x \text{ is the } \text{Gödel number of a theorem of } \text{PA}\]

\[\iff \exists y \text{ Pr}_{\text{PA}}(y, x)\]

Thus (3-8) shows that the truth that \(A\) is a theorem of \(PA\) can be represented as knowledge in \(PA\) since the sentence \(\text{Thm}_{\text{PA}}(*A*)\) expresses this truth can be proved in \(PA\). This shows that \(PA\) is a self-referential system.

(IV) Many systems are not self-referential. Simple material objects are not. Many set theoretical models are so simple that self-reference is impossible. Theories for which Gödel's theorem cannot be proved are not self-referential. An example is Presburger arithmetic.

3.5 REMARK. The examples show that there is a close kinship between self-consciousness and self-reference. In the following, the two will be identified.

3.6 IDEA. My idea is to define \(SR\) as the class of all systems in \(M(\omega)\) which are self-referential. This is a natural class of systems intermediate between \(U\) and \(M(\omega)\). We may therefore try to identify \(\mathcal{\varphi}\) with the theory whose axioms are the set of sentences in \(L\) which are true in all systems in \(SR\). Thus we identify \(\mathcal{\varphi} = \text{Th}(SR)\). It follows that \(SR \subset M(\mathcal{\varphi})\). As shown in Example 3.4(I), the universe \(U\) is self-referential. Hence

\[(\mathcal{U}) \subset SR \subset M(\mathcal{\varphi}) \subset M(\omega)\]

It is clear that \(M(\mathcal{\varphi})\) contains systems which are not elementarily equivalent with \(U\) so that \(\mathcal{\varphi} \neq U\). To decide whether \(\mathcal{\varphi} \neq U\) or \(\mathcal{\varphi} \cong U\), we need to know what sentences are true in all models in \(M(\mathcal{\varphi})\). By Remark 3.5 and the identity \(\mathcal{\varphi} = \text{Th}(SR)\), this is equivalent to asking what follows from the mere fact that \(U\) is a self-conscious/self-referential system. But this is the Cartesian question from § 1.3. Since the identification \(\mathcal{\varphi} = \text{Th}(SR)\) leads to the Cartesian question, this gives a strong indication that the identification is correct.

The empiricist answer to the Cartesian question is that no sentence in \(L\), except those in \(\mathcal{\varphi}\), follows from the mere fact that \(U\) is self-referential. On this view, every sentence in \(\mathcal{\varphi}\) is also true in all models in \(M(\mathcal{\varphi})\). The rationalist answer is the opposite. My own feeling is that self-reference in a system is a special quality that it is unlikely that
this should not make a difference as to what sentences in L are true in all models in SR. This would imply the existence of synthetic a priori truths.

3.7 EXAMPLES. Is it possible to give an example of a synthetic a priori proposition? This has proved to be a difficult question. What should come first, an example of a synthetic a priori proposition or a theory of the synthetic a priori? My own opinion is that the theory should come first. The concept of the synthetic a priori is so subtle that we need a theory to be able to prove that a proposed example is satisfactory. In § 3.6, an idea which may lead to a theory is outlined. This shows that it need not be hopeless to develop a theory of the synthetic a priori without the guidance of a collection of good examples. Nevertheless, here come some proposals.

(I) If there is any synthetic a priori at all, then it would seem that Descartes's proposition Cogito should be an example:

\[(3-9) \quad \text{there is consciousness (in the universe), or}
\]

\[\text{there are thoughts (in the universe).} \]

A possible empiricist objection to this proposal is that (3-9) are existential sentences and therefore they must express empirical truths. This is a misguided objection. Note that there are no unconscious thoughts. Whenever there are thoughts, there is also automatically consciousness and knowledge of them. It is possible to be ignorant of empirical facts but not of the existence of thoughts. This shows that knowledge of thoughts and consciousness has a necessity which empirical knowledge does not. A person's self-referential knowledge of his own thoughts is synthetic a priori. His knowledge of another person's thoughts is synthetic a posteriori. The synthetic a priori depends on the self-reference.

(II) Gödel's incompleteness theorem may be a source of synthetic a priori truths, e.g., the Gödel sentence

\[(3-10) \quad \text{Con}^T_{\text{PA}} \]

and the reflection principles

\[(3-11) \quad \text{Thm}(A') \to A \]

for every sentence A in L(T). T is assumed to be a true axiomatisable extension of PA. The sentences (3-10) and (3-11) are no doubt true; but this insight seems not to be the result of a conceptual analysis. To decide whether the sentences (3-10) and (3-11) really are synthetic a priori, we need a better understanding of the process through which we come to see that the sentences are true.

(III) The axioms of the set theory ZF may give examples of synthetic a priori propositions. One candidate is the axiom of infinity

\[(3-12) \quad \exists x (\emptyset \in x \land \forall y (y \in x \to y \cup \{y\} \in x)) \]

Since the beginning of this century, most logicians and philosophers have felt that axioms of infinity cannot be considered to be analytic. One possibility is that (3-12) may be synthetic a priori. But there are other possibilities also. E.g., it may be possible that (3-12) has no truth-value. Let absolute set theory be ZF with the axiom of infinity deleted. Then we may see ZF as one possible extension of absolute set theory. Another extension is obtained by adding the negation of the axiom of infinity. This would be analogous with the situation in geometry where absolute geometry can be extended to Euclidean as well as to non-Euclidean geometry. To decide the epistemological status of (3-12), we need a theory of the synthetic a priori as well as a better understanding of the process which generates sets.

(IV) Another possible example from set theory is the axiom of choice:

\[(3-13) \quad \text{for every set } x \text{ there exists a function } f : \mathcal{P}(x) \to x \text{ such that for all } y \in \mathcal{P}(x), f(y) \in y.} \]

3.8 REMARK. The basic idea in empiricism is the belief that experience is the only source of information about the universe. Using the ideas proposed in this section, we can formulate the rationalistic view: We have two sources of information about the universe, namely experience and the nonempirical insight that the universe is a self-referential system.

4.4 A Programme

4.1 Programme. (I) Develop the notion of self-referential systems. We need a unified theory which covers both self-referential formal theories and self-referential physical systems like the universe. The class of self-referential systems must be so described that the admissible self-refer-
rental systems can be models of theories since otherwise the class cannot be used to define the theory $\Psi$ of synthetic apriori truths. 

II. Let SR be the class of self-referential systems thus defined. Investigate the theory Th(SR) of sentences true in all systems in SR. Identify $\Psi$ with Th(SR) as suggested in § 3.6, $\Psi = $ Th(SR). If $\Psi = $ Th(SR) = $\lambda$, then empiricism is correct. If $\lambda \subset \Psi = $ Th(SR), then rationalism may be correct. Even with $\lambda \subset \Psi = $ Th(SR), there may be a way of avoiding rationalism, namely if it can be shown that the sentences in Th(SR) - $\lambda$ are all empirical. Whatever the correct solution is to this philosophical problem, if $\lambda \subset $ Th(SR), then the theory Th(SR) appears to be of considerable interest in itself and worthy investigation.

4.2 Logical Rationalism. Once it happened that the amalgamation of the ideas of classical empiricism with logic led to the most powerful form of empiricism so far, namely logical empiricism. We have the equation

\[(4-1) \quad \text{empiricism} + \text{logic} = \text{logical empiricism}\]

Similarly I propose that the amalgamation of the best ideas in classical rationalism, i.e., some of Descartes's ideas, with modern logic may give rationalism a renaissance and create a new and more powerful form of rationalism:

\[(4-2) \quad \text{rationalism} + \text{logic} = \text{logical rationalism}\]

For logical empiricism, formal logic in the object language, i.e., Frege's and Russell's logic, suffices. But for logical rationalism, even mathematical logic is needed. It is clear that the development of a logical rationalism is a much less trivial problem than the development of logical empiricism was.

4-5 Beneficial Consequences

5.1 Introduction. In this section, we consider some possible beneficial consequences of the existence of synthetic apriori propositions. We consider two cases where the empiricist thesis apparently has failed.
The fundamental problem in the foundations of probability theory is the interpretation of the calculus. In particular, concerning probability statements

\[ P(A) = c \]

what do they mean? on what evidence can they be based?

5.4 Salmon's Criteria. Salmon (1967) formulates three criteria of adequacy for interpretations of the probability calculus.

Admissibility: An interpretation of a formal system is admissible if the meanings assigned to the primitive terms by the interpretation transform the formal axioms into true statements.

Assessability: The interpretation must imply that there is a method of assessing values to probability terms.

Applicability: The interpretation must imply that the probability calculus becomes applicable for predictions in the empirical sciences and in practical life.

5.5 Carnap and Reichenbach. Salmon considers a number of interpretations of probability. Those which are interesting here are Carnap's logical interpretation and Reichenbach's frequency interpretation. Both satisfy the admissibility criterion.

In Carnap's logical interpretation, the probability statements (5-1) are analytic. Since the statements are analytic, they are apriori and the assessability condition is satisfied. Once weightings have been assigned to state descriptions, \( P(A) \) can be calculated. But because \( P(A) = c \) is analytic, it contains no information about reality and cannot be applied to reality so that the applicability criterion is not satisfied. To satisfy the assessability criterion, \( P(A) = c \) must be apriori. To satisfy the applicability criterion, \( P(A) = c \) must be synthetic.

In Reichenbach's frequency interpretation, the probability \( P(A) \) is the limit of the relative frequency of the occurrence of the event \( A \) in an infinite sequence of the underlying random experiment. Thus the probability \( P(H) \) of heads in a given coin is the limiting relative frequency of heads in one infinite sequence of flippings with this coin. It follows that probability statements like \( P(A) = c \) in the frequency interpretation are synthetic and aposteriori. Since the statements are synthetic, they satisfy the applicability condition. But the assessability criterion is not satisfied. Since \( P(A) \) is the limit of a potentially infinite empirical series, there is no way to compute \( P(A) \). The frequency interpretation fails to satisfy Assessability because it makes \( P(A) = c \) aposteriori. Again we see that both criteria can be satisfied simultaneously only if \( P(A) = c \) is synthetic apriori.

5.6 Synthetic Apriori Probability. The analysis in § 5.5 indicates that a rationalistic theory of the synthetic apriori as suggested in Section 4 possibly can lead to an interpretation of probability theory which satisfies all three of the Salmon criteria in § 5.4. The assessability criterion demands that a theory of probability should be synthetic; the applicability condition demands that it should be synthetic.

5.7 REMARK. It may turn out that philosophical investigations will show that the idea of synthetic apriori propositions is untenable. E.g., it may turn out that the proposition there is consciousness (in the universe) is just as empirical as 'there are galaxies (in the universe). It is nevertheless still of interest to know whether the theory \( Th(SR) \) of all sentences true in all self-referential systems coincides with the theory \( A \) of analytic truths. It if happens that there is no synthetic apriori, we may shift our attention from the theory \( P \) to the theory \( Th(SR) \). If \( A \subset Th(SR) \subset \Gamma \), then \( Th(SR) \) may be useful in the foundations of mathematics and probability theory even though no sentence in \( Th(SR) \) can be claimed to be synthetic apriori. If \( A \subset Th(SR) \subset \Gamma \), then the study of the systems in \( SR \) and the theory \( Th(SR) \) may develop into a new branch of logic.

5.8 Model Theory. The class of self-referential models is an interesting and natural subclass of the class of all models. Nevertheless, they seem not to have been studied systematically before in model theory. In the standard reference on model theory, Chang and Keisler (1973), they are not mentioned. In the present article, I have pointed out a connection between this class and the epistemology, notably the problem of the synthetic apriori. The study of self-referential systems has given remarkable results in other fields like proof theory and recursion theory. It is highly unlikely that a model theoretic study of self-referential systems should not also give important results. It is amazing that this class of models and their logic have not been systematically studied before.
5.9 OBJECTION (Tom Eriksen). Another possible objection is the following. Some of the self-referential systems in Example 3.4 do not contain any consciousness in the strict sense. Therefore self-referential systems have nothing to do with Descartes’s problem, the Cauterian question, and the synthetic apriori. Descartes’s concern was clearly self-conscious systems, not only self-referential systems, and the classical definition of the synthetic apriori is in terms of self-conscious systems.

ANSWER:
Let SC be the class of all self-conscious systems. Then it appears that SC ⊆ SR so that Th(SR) ⊆ Th(SC). Even if SR ≠ SC, it does not follow from this alone that Th(SC) = Th(SR); we may still have Th(SC) = Th(SR). Thus even in case SR ≠ SC, it is not excluded that the synthetic apriori is the logic of self-referential systems.

Suppose that Th(SC) ≠ Th(SR). If A ≠ Th(SR) ≠ Th(SC), then Th(SR) can be used to prove that there are synthetic apriori truths even though Th(SR) does not exhaust the class Π of synthetic apriori truths. Th(SR) will give a partial characterisation of the synthetic apriori, and we will have made progress towards a complete characterisation. If A = Th(SR) ≠ Th(SC) = ∅, then an examination of SR will leave open the question whether there are synthetic apriori truths. We may then have to attack the possibly deeper and more difficult problem of understanding the nature of consciousness in order to be able to characterise the synthetic apriori. Should it turn out that A = Th(SR) = Th(SC) = ∅, then empiricism will have triumphed.

---

References


5. Russell's Antinomy and Diagonalisation

1.1 The Abstraction Principle. In Frege's and Cantor's systems, the following principle is central:

Let \( A(y) \) be a formula of \( \mathcal{L} = \{ \alpha \} \) having only \( y \) free. Then

\[ \forall x (x \in \{ y | A(y) \}) \leftrightarrow A(x) \]

where \( x \) is free for \( y \) in \( A(y) \).

1.2 Russell's Antinomy. Russell noted that the abstraction principle has as a consequence that

\[ B = \{ x | x \neq x \} \]

is a set and that this implies

\[ B \in B \leftrightarrow B \notin B \]

which is a contradiction.

1.3 PROBLEM. The set \( A \) defined by diagonalisation in the proof of Cantor's theorem satisfies

\[ n \in A \leftrightarrow n \notin A_n \]

We see that there is some similarity with the definition of Russell's class. This suggests that Russell's antinomy may be derived by diagonalisation.

To achieve such a derivation, we first prove a lemma on 2-place relations \( R \) using diagonalisation. The lemma can be proved in ZF, indeed in Zermelo set theory without infinity, and is independent of the abstraction principle of naive set theory.

1.4 LEMMA. Let \( R \) be a binary relation with domain \( D_R \) and codomain \( C_R \) such that \( C_R \subseteq D_R \). Define for every \( y \in D_R \)

\[ S(R, y) = \{ x \in D_R | R(x, y) \} \]

Then there is a set \( A \subseteq D_R \) such that

\[ A \neq S(R, y) \quad \text{for all } y \in D_R \]

PROOF:
Define by diagonalisation

(1) \[ A = \{ x \in D_R | \neg R(x, x) \} \]

Suppose

(2) \[ A = S(R, b) \quad \text{for some } b \in D_R \]

Then

(3) \[ \forall x (x \in D_R \land R(x, b) \leftrightarrow x \in D_R \land \neg R(x, x)) \]

and therefore, taking \( x = b \),

(4) \[ b \in D_R \land R(b, b) \leftrightarrow b \in D_R \land \neg R(b, b) \]

Since, by assumption, \( b \in D_R \), this implies the contradiction

(5) \[ R(b, b) \leftrightarrow \neg R(b, b) \]

1.5 We now derive Russell's paradox from the lemma together with the abstraction principle.

As our universe, we choose the class \( V \) of all sets. Since

\[ V = \{ x | x = x \} \]

\( V \) is itself a set according to the abstraction principle. In the lemma, we let \( R \) be the \( \in \)-relation. Then

\[ D_\in = V, \quad C_\in = V - \{ \emptyset \}, \quad C_\in \subseteq D_\in \]

The set \( A \) becomes for \( R = \in \)

(1*) \[ A = \{ x \in V | x \neq x \} = \{ x | x \notin x \} \]

which is precisely Russell's class. Since

\[ S(\in, y) = \{ x \in V | x \in y \} = y \quad \text{for every } y \]

we have in particular

(2*) \[ A = S(\in, A) \]
We now follow the proof of the lemma line by line. (2*) and (3) give

\( (3^*) \quad \forall x (x \in V \land x \in A \leftrightarrow x \in V \land x \in x) \)

Instantiation in (3*) with \( x = A \) gives

\( (4^*) \quad A \in V \land A \in A \leftrightarrow A \in V \land A \in A \)

Since \( A \in V \), we obtain from (4*)

\( (5^*) \quad A \in A \leftrightarrow A \neq A \)

This is, with inessential variations, the usual derivation of Russell’s antinomy.

1.6 We see that Russell’s antinomy is a special case of Lemma 4, obtained by letting \( \mathcal{R} \) be the \( \epsilon \)-relation and applying the abstraction principle. We see that the derivation of Russell’s antinomy may be conceived as an application of Cantor’s diagonal procedure.

1.7 REMARK. Russell was aware that there is a connection between his own antinomy and Cantor diagonalization. In “My Mental Development” in the Schilpp volume, The Library of Living Philosophers (1946), he writes: “In June 1901, this period of honeymoon delight came to an end. Cantor had a proof that there is no greatest cardinal; in applying this proof to the universal class, I was led to the contradiction of classes that are not members of themselves.” It is not clear whether the connection Russell saw between Cantor’s proof and his own derivation is the same as the one constructed above.

**References**


6. An Upward Skolem Paradox

6-1 Preliminaries

1.1 THEOREM (Cantor). Let \( \kappa \) be the cardinal number of the set of natural numbers. Then \( \kappa < 2^\omega \).

1.2 THEOREM (Downward Löwenheim-Skolem Theorem). Let \( T \) be a countable theory which has an infinite model. Then \( T \) has a countably infinite model.

1.3 THEOREM (Upward Löwenheim-Skolem Theorem). Let \( T \) be a countable theory which has an infinite model of cardinality \( \kappa \). Then \( T \) has models of all cardinalities \( \geq \kappa \).

1.4 ABSTRACT. Skolem’s paradox is exposed and analysed. This paradox is based on the downward Löwenheim-Skolem theorem. An analogous paradox based on the upward Löwenheim-Skolem theorem is formulated and solved. A possible connection with Lindström’s theorem is conjectured.

6-2 The Downward Skolem Paradox

2.1 The Paradox. The following argument is due to Skolem (1922-1923). If \( ZF \) is consistent, it has a model. Such a model must be infinite. By the downward Löwenheim-Skolem theorem, \( ZF \) has a
countably infinite model, \( M = (M, \in) \), i.e., there are only countably many sets in \( M \). On the other hand, there is a theorem in ZF, Cantor's theorem 1.1, which asserts the existence of uncountably many sets and which must be true in \( M \). This appears to be contradictory.

We now give a more careful statement of the paradox and at the same time an analysis of it.

2.2 DEFINITION. Let \( M = (M, \in) \) be a model of ZF.  
(i) \( M \) is a transitive model iff \( \forall x \forall y (x \in y \land y \in M \Rightarrow x \in M) \)  
(ii) \( M \) has the countable bijection property iff for every countably infinite set \( A \in M \) there is a bijection \( f: \alpha_M \rightarrow A \) such that \( f \in M \).  
Here \( \alpha_M \) is the element of \( M \) which represents \( \in \) in \( M \).

2.3 THEOREM. Assume that ZF is consistent. Then ZF has a countable model \( M \). Any such model \( M \) is either not transitive or does not have the countable bijection property.  
PROOF:  
The existence of \( M \) follows by the downward Löwenheim-Skolem theorem. Consider any such model \( M \). Suppose that \( M \) is transitive and has the countable bijection property. We derive a contradiction. Let \( \alpha_M \) and \( \beta(\alpha)_M \) be the representatives in \( M \) of \( \alpha \) and \( \beta(\alpha) \), respectively. Then \( \beta(\alpha)_M \) must be uncountable because otherwise, by the countable bijection property, there is in \( M \) a bijection \( f: \alpha_M \rightarrow \beta(\alpha)_M \). This contradicts the truth in \( M \) of Cantor's theorem. By the transitivity of \( M \),  
\[ \forall x (x \in \beta(\alpha)_M \Rightarrow x \in M) \]  
Then \( M \) is uncountable. This contradicts the consequence of the downward Löwenheim-Skolem theorem that \( M \) is countable.

2.4 THEOREM. Assume that ZF is consistent. Then ZF has a countable model \( M = (M, \in) \). Any such model is isomorphic with a countable model \( M^* \) which does not satisfy the countable bijection property.  
PROOF:  
The existence of \( M \) is guaranteed by the downward Löwenheim-Skolem theorem. Since we have not assumed that \( M \) is transitive, it may be possible to have that \( \alpha_M \) is countable, \( \beta(\alpha)_M \) is uncountable but has only countably many elements in common with \( M \), and \( M \) has the countable bijection property. We now obtain another model \( M^\ast = (M^\ast, \in^\ast) \). Define  
\[ x^\ast = \{ y \in M \mid y \models x \}, \text{ for each } x \in M \]  
\[ M^\ast = \{ x^\ast \mid x \in M \} \]  
\[ x^\ast = y^\ast \iff x \equiv y, \text{ for all } x^\ast, y^\ast \in M^\ast \]  
\( M^\ast \) is clearly countable as is every \( x^\ast \in M^\ast \), and the mapping \( x \rightarrow x^\ast \) is a homomorphism of \( M \) and \( M^\ast \). We show that the mapping is injective. Assume to the contrary that there are \( x, y \in M \) such that  
\[ (2-1) \quad x^\ast = y^\ast, \text{ and} \]  
\[ (2-2) \quad x \neq y \]  
Since \( M \) satisfies the extensionality axiom, it follows from (2-2) that there is a \( y_0 \in M \) such that  
\[ (2-3) \quad y_0 \in x, \quad y_0 \notin y, \text{ or} \]  
\[ (2-4) \quad y_0 \in y, \quad y_0 \notin x \]  
Without loss of generality, we consider only possibility (2-3). From (2-1) follows  
\[ (2-5) \quad \forall y (y \in M \land y \in x \Rightarrow y \in M \land y \in x) \]  
Hence  
\[ (2-6) \quad y_0 \in M \land y_0 \in x \Rightarrow y_0 \in M \land y_0 \in x \]  
Since \( y_0 \in M \),  
\[ (2-7) \quad y_0 \in x \Rightarrow y_0 \in x \]  
which is incompatible with (2-3). It follows that the mapping \( x \rightarrow x^\ast \) is an isomorphism of \( M \) and \( M^\ast \).  
Suppose \( M^\ast \) has the countable bijection property. Let \( \alpha_{M^\ast} \) and \( \beta(\alpha)_{M^\ast} \) represent \( \alpha \) and \( \beta(\alpha) \) in \( M^\ast \). As shown, \( \alpha_{M^\ast} \) and \( \beta(\alpha)_{M^\ast} \) are countably infinite. By the countable bijection property, there is an \( f \in M^\ast \) such that \( f: \alpha_{M^\ast} \rightarrow \beta(\alpha)_{M^\ast} \) is injective. It follows that Cantor's theorem fails in \( M^\ast \). But this is impossible because  
\[ M^\ast \models ZF \]  
since \( M^\ast \) and \( M \) are isomorphic.
2.5 REMARKS. (I) Unfortunately, most expositions of Skolem's paradox are somewhat sloppy. They take it for granted that the model $M$ given by the downward Löwenheim-Skolem theorem is transitive, or else they fail to move to the model $M^*$. Such deficient derivations occur in Stoll (1961), Shoenfield (1967), Hatcher (1981), and even in Skolem (1923). A notable exception is Bell and Machover (1977). They make the move from $M$ to $M^*$ as in the proof of Theorem 2.4.

(II) Theorem 2.4 gives an intuitive solution to the downward Skolem paradox. From outside the model $M^*$, it is possible to see that $P(n)_{M^*}$ is countable. From inside $M^*$, it is impossible to see that $P(n)_{M^*}$ is countable since no bijection between $n_{M^*}$ and $P(n)_{M^*}$ exists in the model.

6.3 The Upward Skolem Paradox

3.1 PROBLEM. Skolem's paradox exposed in the preceding section is based on the downward Löwenheim-Skolem theorem. Is it possible to construct a paradox closely analogous to Skolem's but based on the upward Löwenheim-Skolem theorem? In the present section, I give an affirmative answer to this question.

3.2 DEFINITION. We define some set theorems in the language $L = \{ \in \}$. As our starting point, we take an axiomatisation of ZF, e.g., as exposed in Hatcher (1982).

(I) Zermelo set theory $Z$ is $ZF$ without the axiom of replacement but with the Axiom of Separation. Thus $Z$ has the axioms ZF.1-ZF.8 in Hatcher (1982).

(II) Absolute set theory $S$ is Zermelo set theory but with the axiom of infinity

$ZF.8 \quad \exists x (\emptyset \in x \land \forall y (y \in x \rightarrow y' \in x))$

deleted. Here $y' = y \cup \{ y \}$ is the successor of $y$. $S$ has the axioms ZF.1-ZF.7 in Hatcher (1982).

(III) Set theoretic arithmetic $SA$ is Zermelo set theory but with the axiom of infinity replaced by its own negation. Thus the axioms of $SA$ are ZF.1-ZF.7 and

$SA.8 \quad \neg \exists x (\emptyset \in x \land \forall y (y \in x \rightarrow y' \in x))$

3.3 REMARK. SA is equivalent to Peano arithmetic PA in the sense that each of the two theories has an interpretation in the other. SA also has a standard model which will be defined in the next paragraphs. More information about SA can be found in Chang & Keisler (1972).

3.4 DEFINITION. The class of hereditarily finite sets (HF sets) is defined inductively as the smallest set satisfying

(1) $\emptyset$ is hereditarily finite.

(2) $x$ is finite and $\forall y (y \in x \rightarrow y$ is hereditarily finite $)$ $\Rightarrow x$ is hereditarily finite.

(*) A HF set is a finite set which only yields finite sets when it is unpacked."

3.5 DEFINITION. The set $V_{\omega}$ is defined inductively by

$V_{\emptyset} = \emptyset$

$V_{x+1} = \mathcal{P}(V_x)$

$V_{\omega} = \bigcup_{x \in \omega} V_x$

3.6 LEMMA. For any set $A$, $A$ is hereditarily finite $\iff A \in V_{\omega}$

3.7 DEFINITION. The standard model of SA is the model $\mathcal{N} = (V_{\omega} , \in)$

3.8 REMARK. By the lemma, the standard model consists of all hereditarily finite sets. To show that $\mathcal{N}$ is a model of SA, verify SA.1-SA.8 in $\mathcal{N}$.

3.9 LEMMA. $\mathcal{N}$ is an infinite and countable model. There are only countably many hereditarily finite sets.
3.10 Assumption I. To derive the paradox, we need the following interpretation of $ZF_{8}$ and $SA_{8}$.

The infinity axiom $ZF_{8}$ expresses exactly that there is an infinite set.

The negation $SA_{8}$ of the infinity axiom expresses exactly that there is no infinite set.

3.11 The Paradox. Since $\mathcal{N} \models SA$, $SA$ has a countably infinite model. By upward Löwenheim-Skolem, $SA$ has an uncountable model $\mathcal{M} = (M, e)$. Since there are only countably many HF sets, $M$ contains a set $A$ which is not HF. Thus either $A$ is infinite or contains inside it an infinite set.

On the other hand, $SA$ has an axiom $SA_{8}$ which by Assumption I expresses that there is no infinite set. This contradicts the consequence of the upward Löwenheim-Skolem paradox that $\mathcal{M} \models SA$.

We make a first attempt to solve the paradox using the notion of a transitive model of Definition 2.2.

3.12 Lemma. The standard model $\mathcal{N} = (V_{\alpha}, e)$ is transitive.

Proof:
Let $x \in y, y \in V_{\alpha}$. Let $V_{\alpha+1}$ be the first set in the $\alpha$-hierarchy where $y$ occurs. Then $x \in V_{\alpha}$ so that $x \in V_{\alpha}$.

3.13 An Attempt. The idea is that though $SA$ has models $\mathcal{M} = (M, e)$ containing non-HF sets, all such models are non-transitive. $M$ itself contains only finite sets. An infinite set buried in a non-HF set need not be and is not a member of $M$. Therefore it is possible to have $SA_{8}$, in the interpretation of Assumption I, true in $\mathcal{M}$.

This attempt does not work. The $\cup$-operation is definable in $S$ and therefore also in $SA$:

$$\forall x \forall y \ (y \in \cup x \leftrightarrow \exists z \ (x = z \land y \in z))$$

We can also define in $SA$ the $n$th iteration of the $\cup$-operation:

$$\cup(0)x = x$$

$$\cup(n+1)x = \cup\cup(n)x$$

Let $\mathcal{M} = (M, e)$ be a model of $SA$. If $x \in M$, then $\cup(n)x \in M$, for all $n$. This follows since $\cup(n)$ is definable in $SA$. In particular, if $x \in M$ is non-HF, then $\cup(n)x \in M$ is infinite for some $n$ and $\cup(n)x \in M$. For let $x$

be non-HF and $y$ be an infinite set buried in $x$. Then for some $n$, $y \subseteq \cup(n)x \in M$. Therefore $\cup(n)x$ must be infinite since it has an infinite subset. It does not help if by non-transitivity $y \in M$. It follows that every model of $SA$, which contains a non-HF set, contains an infinite set. We have the following theorem.

3.14 Theorem. All models of $SA$, except the standard model $\mathcal{N}$, contain an infinite set. Moreover, $SA$ has countable models containing an infinite set.

Proof:
The first half of the theorem was proved in the preceding paragraph. Two distinct proofs are given of the second half of the theorem.

(I) Suppose that every countable model of $SA$ is isomorphic with $\mathcal{N}$ by the Löwenheim-Skolem categoricity theorem, $SA$ is complete. This contradicts Gödel's incompleteness theorem. Any such countable nonstandard model $\mathcal{M}$ has $\mathcal{N}$ as a proper submodel and therefore must contain a non-HF set. But then $\mathcal{M}$ contains an infinite set as shown.

(II) In the alternative proof, we use an analogue of Skolem's method of defining a nonstandard model of $PA$. We expand the language with a new constant $c$, $\mathcal{L} = (e, c)$. Every HF set can be defined in $SA$ using terms built only from pairs of set brackets $[\ ]$. We call such terms bracket terms. Let $T$ be the theory in $\mathcal{L}$ having as axioms all the axioms of $SA$ together with the set of sentences:

$$[e \neq \text{it is a bracket term}]$$

By the compactness theorem, $T$ has a model and, by the downward Löwenheim-Skolem theorem, a countable model $\mathcal{M}_{c} = (M, e, c)$. Then $M$ contains a non-HF set $e$. Let $\mathcal{M} = (M, e)$ be the restriction of $\mathcal{M}_{c}$ to $\mathcal{L} = \{e\}$. Then $\mathcal{M}$ is a countable nonstandard model of $SA$.

3.15 Remark. Note that Part (II) of the proof implies that essentially the same paradox as in § 3.11 can be derived using downward Löwenheim-Skolem together with the compactness theorem, instead of upward Löwenheim-Skolem.

3.16 The Solution. The theorem shows that the only possible solution to the upward Skolem paradox is to deny Assumption I. In contrast to a widespread opinion, $ZF_{8}$ does not exactly express that there is an infi-
nite set and SA.8 does not exactly express that there is no infinite set. The infinity axiom

\[ \exists x (\emptyset \notin x \land \forall y (y \in x \rightarrow y' \in x)) \]

expresses more than just the existence of an infinite set. It says that there is an infinite set and this set contains \( \emptyset \) and is closed under the \('' \) operation. Therefore the negation of ZF.8

\[ \neg \exists x (\emptyset \in x \land \forall y (y \in x \rightarrow y' \in x)) \]

is weaker than the statement "There is no infinite set". SA.8 says only that there is no set which contains \( \emptyset \) and is closed under the \('' \) operation. This condition is satisfied by all HF-sets but it is also satisfied by many infinite sets.

3.17 THEOREM. (I) There is no sentence in any elementary first-order language \( \mathcal{L} = \{e, \ldots\} \) where \( e \) satisfies the axioms of S, which expresses exactly that there is an infinite set. There is no sentence which expresses exactly that there is no infinite set.

(I) There is an infinite set of sentences in the language \( \mathcal{L} = \{e, c\} \) which expresses exactly that there is an infinite set.

(II) There is no set of sentences in any elementary first-order language \( \mathcal{L} = \{e, \ldots\} \) where \( e \) satisfies the axioms of S, which expresses exactly that there is no infinite set.

(IV) In non-elementary first-order logic, it is possible to express exactly in one sentence that there is no infinite set.

PROOF:

(I) has already been proved.

(II) As the infinite set of sentences we take the axioms of S together with the set \[ \{c \equiv \text{false if } t \text{ is a bracket term}\} \]
defined in the proof of Theorem 3.14.

(III) Suppose there is such a set. Let T be the theory which has this set of sentences together with the axioms of S as its nonlogical axioms. Then T has, just as was shown for SA, a model containing an infinite set which contradicts the choice of T.

(IV) Let \( x = y \) denote that \( x \) and \( y \) are sets of the same cardinality. This relation is definable in S. We now operate in the logic \( L_{\mathcal{E}1(\mathcal{E})} \) which allows countable conjunctions and disjunctions. The sentence

\[ \forall x (x = 0 \lor x = 1 \lor x = 2 \lor \ldots) \]

expresses exactly that every set is finite.

### 6-4 Lindström's Theorem

4.1 Lindström (1966, 1969) defines a very general class of logical systems. It includes the usual lower predicate calculus \( L_{\mathcal{E}1(\mathcal{E})} \), higher order logics, logics with generalised quantifiers, and logics with infinite conjunctions and disjunctions like \( L_{\mathcal{E}1(\mathcal{E})} \) and \( L_{\mathcal{E}1(\mathcal{E})} \). It is then possible to prove the following theorem, cf. Barwise (1974).

4.2 THEOREM (Lindström). Let \( L \) be a logic which satisfies the downward Löwenheim-Skolem theorem and either the upward Löwenheim-Skolem theorem or the compactness theorem. Then \( L \) is the usual lower predicate calculus \( L_{\mathcal{E}1(\mathcal{E})} \).

4.3 CONJECTURE. There are still some open problems connected with the understanding of Lindström's theorem. Hao Wang has, e.g., in a survey article asked why the Löwenheim-Skolem theorems have such an apparently central role in understanding the nature of logic. It is possible that the downward and upward Skolem paradoxes may throw some light on this question.

Note first that the condition used by Lindström-Barwise to characterise the lower predicate calculus is precisely the satisfaction of the theorems which were central in the derivations above of the two paradoxes. Skolem's paradox and Theorem 2.4 are based on the downward Löwenheim-Skolem theorem. The new paradox developed in Section 3 and Theorem 3.14 can be based on the upward Löwenheim-Skolem theorem or on the compactness theorem as shown. We may also note that set theory has a close relation to logic. By the completeness theorem, a sentence is logically true iff it is true in all set theoretic models. In view of this, we make the casual conjecture that a closer comparison of the two Skolem paradoxes with Lindström's theorem may give new insights into the role played by the Löwenheim-Skolem theorems in Lindström's theorem. The idea is that among the logics considered, \( L_{\mathcal{E}1(\mathcal{E})} \) is the only
strictly finitistic logic. Therefore \( L_{\infty\omega} \) is the only one which neither has resources to express a clear distinction between different infinite cardinalities, as brought out by the downward Skolem paradox, nor resources to express a clear distinction between finite and infinite, as brought out by the upward Skolem paradox.

### 4.4 Standard Models

Finally we observe that the upward Skolem paradox throws some light on the notion of a standard model. Some theories like PA and SA have standard models. Other theories like group theory and the theory of Boolean algebras do not. A standard model is an intended model. The standard model is determined by the intentions of the author of the theory and not by the logical properties of the theory. SA illustrates this nicely. All of its models except the standard model have the property which the characteristic SA-axioms is intended to exclude, namely the property of containing an infinite set.

### Notes

Section 2. The proof of Theorem 2.4 is adapted from Bell and Machover (1977).

Section 3. A paradox akin to the upward Skolem paradox is given by Shoenfield (1967). His version of the paradox is based on a comparison between first-order and second-order Peano arithmetic. The upward Skolem paradox, as given in the present chapter, stays entirely within the realm of first-order theories.

### References


7. Questions and Results on ω-Consistency

7-1 An ω-Inconsistent Extension of PA

1.1 NOTATION. We use PA to designate Peano arithmetic. We operate in the language $L = \{0, +, \cdot \}$ of PA. $\mathcal{N} = (\mathbb{N}, 0, +, \cdot)$ is the standard model of PA.

1.2 DEFINITION. Let $T$ be a theory in $L$. $T$ is $\omega$-consistent $\iff$ there is no formula $A(x)$ of $L$ having only $x$ free such that $T \models A(0), T \models A(1), \ldots$, and $T \models \exists x \neg A(x)$

A theory which is not $\omega$-consistent is $\omega$-inconsistent.

1.3 LEMMA. If $\mathcal{N} \models T$, then $T$ is $\omega$-consistent.

1.4 LEMMA. If $T$ is $\omega$-consistent, then $T$ is consistent.

1.5 THEOREM (Gödel's First Incompleteness Theorem). Let $T$ be an axiomatisable extension of PA. Then there is a sentence $\forall x \ G(x)$ which asserts its own unprovability in $T$ and satisfies:

(I) If $T$ is consistent, then $T \vdash \forall x \ G(x)$

(II) If $T$ is $\omega$-consistent, then $T \vdash \neg \forall x \ G(x)$

1.6 PROBLEM. Rosser showed how the statement (II) in Gödel's theorem can be strengthened by substituting 'consistency' for 'ω-consistency'. The concept of $\omega$-consistency is, however, of interest in its own right. By Lemma 1.3, PA is $\omega$-consistent. Can we find an extension of PA which is consistent but not $\omega$-consistent?

1.7 THEOREM. Let $T$ be an axiomatisable extension of PA such that $\mathcal{N} \models T$. Let $\forall x \ G(x)$ be the Gödel sentence which asserts its own unprovability in $T$. Then

(I) $T + \forall x \ G(x)$ is $\omega$-consistent.

(II) $T + \neg \forall x \ G(x)$ is consistent but not $\omega$-consistent.

PROOF:

(I) Let $P_T(m,n)$ be the relation in $\mathcal{N}$.

$P_T(m,n) \iff n$ is the Gödel number of a proof in $T$ whose last formula has Gödel number $m$.

This relation is decidable and therefore recursive and representable in $T$ by a formula $P_T(x,y)$, i.e.,

(1-1) $P_T(m,n) \iff T \models P_T(m,n)$ for all $m, n$

(1-2) $\neg P_T(m,n) \iff T \models \neg P_T(m,n)$ for all $m, n$

By the fixed-point lemma, there is a formula $\Psi$ such that $T \models \Psi \iff \forall x \neg P_T(x, \ast \Psi^\ast)$

where $\ast \Psi^\ast$ denotes the Gödel number of $\Psi$. Thus $\Psi$ expresses its own unprovability, and we may identify $\forall x \ G(x)$ with $\neg P_T(x, \ast \Psi^\ast)$ and $\forall x \ G(x)$ with $\Psi$. By Gödel's theorem, $T \models \Psi$. Hence not $P_T(n, \ast \Psi^\ast)$ for all $n$. By (1-2),

(1-3) $T \models \neg P_T(n, \ast \Psi^\ast)$ for all $n$

Then in turn

$\mathcal{N} \models \neg P_T(n, \ast \Psi^\ast)$ for all $n$

$\mathcal{N} \models \forall x \neg P_T(x, \ast \Psi^\ast)$

It follows that

$\mathcal{N} \models T + \forall x \ G(x)$

By Lemma 1.3, $T + \forall x \ G(x)$ is $\omega$-consistent.

(II) $T + \neg \forall x \ G(x)$ is consistent since otherwise $T \models \forall x \ G(x)$ which contradicts Gödel's theorem. From (1-3) and $G(x) = \neg P_T(x, \ast \Psi^\ast)$, we have

(1-4) $T \models G(n)$ for all $n$
By predicate logic,
\[ T + \neg \forall x \ G(x) \models \exists x \neg G(x) \]
which together with (1-4) shows that \( T + \neg \forall x \ G(x) \) is \( \omega \)-inconsistent.

7-2 Other Results on \( \omega \)-Consistency

2.1 DEFINITION. (I) The \( \omega \)-rule is the nonfinitary rule
\[ \vdash A(0), \vdash A(1), \ldots \quad (\omega \text{-rule}) \]
\[ \vdash \forall x A(x) \]

(II) \( T \) is \( \omega \)-complete if the \( \omega \)-rule is satisfied by \( T \).

2.2 DEFINITION. (I) The pure computational arithmetic \( S_0 \) is the theory whose nonlogical axioms are all true sentences \( t = u \) and all true sentences \( t \neq u \) where \( t \) and \( u \) are variable-free terms.

(II) \( \omega \)-arithmetic \( S_{0\omega} \) is the theory \( S_0 \) augmented with the \( \omega \)-rule.

2.3 LEMMA. (I) There are \( \omega \)-complete theories which are not \( \omega \)-consistent.

(II) If \( T \) is \( \omega \)-consistent, then \( T \) is consistent.

(III) There are consistent and complete theories which are neither \( \omega \)-consistent nor \( \omega \)-complete.

(IV) If \( T \) is consistent and \( \omega \)-complete, then \( T \) is \( \omega \)-consistent.

(V) If \( T \) is complete and \( \omega \)-consistent, then \( T \) is \( \omega \)-complete.

(VI) There are complete and \( \omega \)-complete theories which are not consistent.

(VII) There are \( \omega \)-consistent and \( \omega \)-complete theories which are not complete.

PROOF:

(I) Let \( T \) be the inconsistent theory.

(II) By Theorem \( 1.7 \), \( PA + \neg \forall x \ G(x) \) is consistent and \( \omega \)-inconsistent.

Let \( T \) be a maximal consistent extension of \( PA + \neg \forall x \ G(x) \).

(IV) Assume that \( T \) is \( \omega \)-inconsistent. Then

\[ (2-1) \quad T \models A(0), \ T \models A(1), \ldots , \ T \models \exists x \neg A(x) \]

By \( \omega \)-completeness,
\[ T \models \forall x A(x) \]
which is incompatible with (2-1) and the consistency of \( T \).

(V) Assume
\[ T \models A(0), \ T \models A(1), \ldots \]
By \( \omega \)-consistency
\[ T \models \exists x \neg A(x) \]
By completeness,
\[ T \models \forall x A(x) \]

(VII) Let \( T_0 \) be the theory with no nonlogical axioms. Let \( T_{0\omega} \) be \( T_0 \) augmented with the \( \omega \)-rule and let \( T \) be the theory whose axioms are all sentences which are provable in \( T_{0\omega} \). Then \( T \) is certainly \( \omega \)-complete and, by Lemma 1.3, \( T \) is \( \omega \)-consistent. To see that \( T \) is incomplete, consider the model \( \mathcal{M} = (\{0\}, \ 0, \ +, \ ) \). We have
\[ \mathcal{M} \models T_0 \]

Let \( T_1 = \text{Th}(\mathcal{M}) \). Then \( T_1 \) is complete. \( T_1 \) is also \( \omega \)-consistent for suppose to the contrary that there is a formula \( B(x) \) such that
\[ T_1 \models B(0), \ T_1 \models B(1), \ldots , \ T_1 \models \exists x \neg B(x) \]

Then
\[ (2-2) \quad \mathcal{M} \models B(0) \]
\[ (2-3) \quad \mathcal{M} \models \exists x \neg B(x) \]

Since \( M = \{0\} \), (2) implies
\[ \mathcal{M} \models \neg B(0) \]

which is incompatible with (2-2). By Lemma 2.3(V), \( T_1 \) is \( \omega \)-complete. But in \( T_1 \) we have
\[ T_1 \models 0 = 0' \]

On the other hand, we have
\[ \text{Th}(\mathcal{M}) \models \neg 0 = 0' \]

Since both \( T \subseteq T_1 \) and \( T \subseteq \text{Th}(\mathcal{M}) \), \( 0 = 0' \) is undecidable in \( T \), and \( T \) is incomplete.
2.4 THEOREM. (I) $S_0$ is axiomatisable.
(II) $S_0 \subseteq PA$
(III) $S_{0#} = Th(\mathcal{N})$

PROOF:
(I) The set of all true sentences $t = u$, where $t, u$ are variable-free terms, is decidable. This follows since $t$ and $u$ are built up from $0$ by the function symbols $'$, $+$ and $\cdot$. Since these functions are recursive, it is always possible to compute the value of $t$ and $u$ and decide the truth of $t = u$. Similarly, the set of all sentences $t = u$ is decidable.

(II) Since all recursive functions are representable in PA, the successor, addition, and multiplication functions, in particular, are representable in PA. This implies that the computations of $t$ and $u$ in $t = u$ or $t \neq u$ can always be done in PA. Therefore, $S_0 \subseteq PA$.

(III) Since $S_0 \subseteq PA \subseteq Th(\mathcal{N})$, and $Th(\mathcal{N})$ is $\omega$-complete, $S_{0#} \subseteq Th(\mathcal{N})$.

To show $S_{0#} = Th(\mathcal{N})$, we prove that $S_{0#}$ is complete. We show that all sentences in prefix form which are true in $\mathcal{N}$ are also theorems of $S_{0#}$. The proof is by induction over the number of quantifiers in the prefix. If there are $n$ quantifiers in the prefix of a PNF formula, we say that the formula is prenex-$n$.

If $A$ is a true closed atomic formula $t = u$, then $A$ is an axiom of $S_{0#}$. Thus $S_{0#} = A$. Similarly, if $A$ is a true sentence $t = u$, $A$ is an axiom of $S_{0#}$. If $A$ is any prenex-$0$ sentence true in $\mathcal{N}$, then $A$ is a truth function of sentences of the form $t = u$. Since the truth value of $t = u$ is decidable in $S_0$, so is the truth value of $A$. Hence $S_{0#} = A$.

Assume $S_{0#}$ contains all true prenex-$n$ sentences. Let

$$\mathcal{N} \models \exists x \ A(x)$$

where $\exists x \ A(x)$ is prenex-$(n+1)$. Then for some $k$, $\mathcal{N} \models A(k)$. Since $A(k)$ is prenex-$n$, $S_{0#} = A(k)$. By predicate logic,

$$S_{0#} = \exists x \ A(x)$$

Let

$$\mathcal{N} \models \forall x \ A(x)$$

with $\forall x \ A(x)$ prenex-$(n+1)$. Then

$$\mathcal{N} \models A(0), \mathcal{N} \models A(1), \ldots$$

By the induction hypothesis,

$$S_{0#} = A(0), S_{0#} = A(1), \ldots$$

By the $\omega$-rule,

$$S_{0#} = \forall x \ A(x)$$

2.5 REMARK. $S_0$ contains all correct arithmetic computations and all the basic atomic facts of arithmetic. It contains no universal arithmetical laws. E.g., simple universal truths like

$$\forall x \ x + 0 = x$$

$$\forall x \forall y \ x + y = y + x$$

are not provable in $S_0$. Nevertheless, in spite of its meagerness, $S_0$ is a sufficient basis for complete number theory if the $\omega$-rule is added. The fact that the addition of the $\omega$-rule to $S_0$ or PA is sufficient to yield complete arithmetic helps to relieve the mystifying effects of Gödel's incompleteness theorem.

2.6 REMARK. We can define a sequence $T_{\alpha}$ of theories, where $\alpha$ is an ordinal. Let

$$T_0 = PA$$

$$T_1 = T_0 + \forall x \ G_0(x)$$

where $\forall x \ G_0(x)$ is the Gödel sentence expressing its own unprovability in $T_0$. Similarly, we define $T_2, T_3, \ldots$ Now consider the theory $T_{0#}$

$$T_{0#} = \cup_{\beta < \alpha} T_{\beta}$$

As axioms of $T_{0#}$ we may take the axioms of $T_0$ and all the sentences $\forall x \ G_{\alpha}(x)$. This set is clearly recursively enumerable (RE). Then $T_{0#}$ is RE. Therefore $T_{0#}$ is axiomatisable. We may therefore continue the process and get

$$T_{0#}, T_{0#}+1, T_{0#}+2, \ldots$$

2.7 THEOREM. Define the sequence $\{T_{\alpha}\}$ as follows.

$$T_0 = PA$$

$$T_{\alpha+1} = \begin{cases} T_{\alpha} + \forall x \ G_{\alpha}(x) & \text{if } T_{\alpha} \text{ is axiomatisable} \\ T_{\alpha} \text{ otherwise} \end{cases}$$

$$T_{\alpha} = \cup_{\beta < \alpha} T_{\beta} \quad \text{if } \alpha \text{ is a limit ordinal}$$
Then $T_\omega$ is not axiomatisable and there is a countable limit ordinal $\gamma$ such that

$$T_\omega = T_\gamma \subseteq \text{Th}(N).$$

2.8 DISCUSSION. The theorem can be proved directly or it follows from a theorem in Turing (1939). Actually, Turing proves a much stronger result. He shows that if we to each $T_\alpha$, adds the class $R_{\alpha+1}$ of all set principles of $T_\alpha$, then $T_\alpha$ is essentially more complete than PA since all sentences in an important subclass of $\text{Th}(N)$ can be proved in $T_\alpha$.

For every $T_\alpha$ in the sequence $\{T_\alpha\}$ in Theorem 2.7, we have two options. We may add $\forall x \, G_\alpha(x)$, or we may add $\neg \forall x \, G_\alpha(x)$ as a new axiom. The argument of Theorem 1.7(i) shows that if we want a true $T_\alpha$, we must choose $\forall x \, G_\alpha(x)$. Some sensitive persons dislike talk about truth in mathematical theories. Theorem 1.7(ii) shows, however, that if we add just one $T_\alpha$ add $\neg \forall x \, G_\alpha(x)$, then we get an $\omega$-inconsistent theory. Thus, if we want an $\omega$-consistent arithmetic, we must add the Gödel sentence rather than its negation. This is a purely syntactical criterion without any reference to truth.

$\omega$-consistency is a desirable feature in an arithmetical theory. It is also desirable that the theory has a complete extension which remains $\omega$-consistent. But these two conditions can only be satisfied by theories $T$ such that $S_\alpha \subseteq T \subseteq \text{Th}(N)$ as shown in the next lemma.

We see that as soon as the axiomatic development of arithmetic has reached a level where Gödel's theorem is provable, then we have no longer any choice for the further axiomatic development of the theory unless we are willing to abstain from $\omega$-consistency.

2.9 LEMMA. Let $T$ be a theory such that

(i) $S_\alpha \subseteq T$

(ii) $T$ is $\omega$-consistent and has a complete $\omega$-consistent extension.

Then $T \subseteq \text{Th}(N)$.

PROOF: Suppose that $T$ satisfies the two conditions, but $T \nless \text{Th}(N)$. Let $T_1$ be a complete and $\omega$-consistent extension of $T$. By Lemma 2.3, $T_1$ is $\omega$-complete. By Theorem 2.4, $T_1 = S_\alpha = \text{Th}(N)$ since $S_\alpha \subseteq T \subseteq T_1$. But this contradicts $T \nless \text{Th}(N)$.

2.10 CONJECTURES. In view of the discussion in §2.8, we may ask about the relationship between $\omega$-consistency and arithmetical truth. To what extent and under what conditions, if any, does $\omega$-consistency force truth upon an arithmetical theory? In an attempt to attack the problem, we state it in the form of three conjectures.

(i) Let $S$ be any theory in $L$.

$$S \text{ is } \omega\text{-consistent } \iff \mathcal{N} \models S$$

(ii) Assume $S_\alpha \subseteq S$. Then

$$S \text{ is } \omega\text{-consistent } \iff \mathcal{N} \models S$$

(iii) Assume $PA \subseteq S$. Then

$$S \text{ is } \omega\text{-consistent } \iff \mathcal{N} \models S$$

The equivalence

$$S \text{ is } \omega\text{-consistent } \iff \mathcal{N} \models S$$

is very strong since it equates $\omega$-consistency and arithmetical truth. The proof for the direction $\iff$ is trivial though since this is just Lemma 1.3. We now examine the direction $\iff$ for the three conjectures.

2.11 On Conjecture 2.10(i). Conjecture 2.10(i) is false as can be seen from the following counterexample. Let

$$\mathcal{M} = (\{0\}, 0, \ast, +, \cdot)$$

$$S = \text{Th}(\mathcal{M})$$

$S$ is the same theory as $T_1$ which was shown to be $\omega$-consistent in the proof of Lemma 2.3(VII). Since

$$\mathcal{N} \models 0 = 0'$$

we have

$$\mathcal{N} \models S$$

2.12 On Conjecture 2.10(ii). To prove Conjecture 2.10(ii), we must be able to prove that for any sentence $A$ in PNF,

$$S \models A \iff \mathcal{N} \models A$$

The implication (*) can be proved for all prenex-0 and prenex-1 sentences $A$. It can also be proved for all prenex-2 sentences which have
It is clear that $\mathcal{N} \models S$. I now indicate how it can be seen that $S$ is $\omega$-inconsistent.

Define

$$S_1 = S_0 + [a + b = b + a]$$

where $a$ and $b$ are new constants. (We actually use the method of Skolem functions.) Clearly, $S_2 \models S_1$. If $S_2$ is $\omega$-consistent, so is $S_1$. Assume $S$ $\omega$-inconsistent. Then there is a formula $A(x)$ of $L_2$ such that

$$(2-4) \quad S \models A(0), \quad S \models A(1), \quad \ldots, \quad S \models \exists x \neg A(x)$$

Hence

$$(2-5) \quad S_1 \models A(0), \quad S_1 \models A(1), \quad \ldots, \quad S_1 \models \exists x \neg A(x)$$

Since all axioms of $S_1$ are variable-free, we can deduce in $S_1$ for some variable-free term $t$ of $L_2 = L \cup \{a, b\}$

$$(2-6) \quad S_1 \models \neg A(t)$$

Then both $a$ and $b$ must occur in $t$ which we show now. Assume without loss of generality that, e.g., $b$ does not occur in $t = t(a)$. Then in turn

$$S_2 \models a+b = b+a \rightarrow \neg A(t(a))$$
$$S_2 \models A(t(a)) \rightarrow a+b = b+a$$

In the deduction in $S_1$ of $A(t(a)) \rightarrow a+b = b+a$, replace all occurrences of $a$ and $b$ by new variables $x$ and $y$, respectively, not occurring in the deduction. Then in turn, by predicate logic,

$$S_2 \models A(t(x)) \rightarrow x+y = y+x$$
$$S_2 \models \forall x \forall y (A(t(x)) \rightarrow x+y = y+x)$$
$$S_2 \models \forall x (A(t(x)) \rightarrow \forall y x+y = y+x)$$
$$S_2 \models A(t(0)) \rightarrow \forall y 0+y = y+0$$

Since $t(0)$ is a variable-free term of $L_1$, it can be computed in $S_0$ and we have for some $n$,

$$S_0 \models t(0) = n$$

Hence

$$S_1 \models A(n) \rightarrow \forall y 0+y = y+0$$

By $(2-5)$,

$$S_1 \models \forall y 0+y = y+0$$

This contradicts

$$S_1 \models \neg A(t)$$

which can easily be shown by a model. Therefore $t = t(a, b)$ must contain both $a$ and $b$. From $(2-5)$ and $(2-6)$ we get

$$(2-7) \quad S_1 \models A(0), \quad S_1 \models A(1), \quad \ldots, \quad S_1 \models \neg A(t(a, b))$$

Consider any deduction in $S_1$ of $\neg A(t(a, b))$. The only deduction rules which can be meaningfully applied in the deduction are the identity axiom (Id), the substitution rule (Subst), the rule of existential introduction ($\exists I$), the tautology rule (Taut).

Given the axioms of $S_1$, we see that these rules leave $a+b$ and $b+a$ unchanged. Thus $t(a, b)$ is $a+b$ or $b+a$. We assume without loss of generality that $t(a, b)$ is $a+b$. From $(2-7)$,

$$(2-8) \quad S_1 \models A(0), \quad S_1 \models A(1), \quad \ldots, \quad S_1 \models \neg A(a+b)$$

Thus there is some property $A(x)$ which $0, 1, \ldots$ all have and which provably fails for $a+b$. But from the axioms of $S_1$, we see that the only properties which $a+b$ provably does not have are

$$x = b+a$$
$$\forall x x = x$$
$$\forall y \forall z x = y+z$$

For some of these properties can

$$S_1 \models A(n)$$

imply that $n$ has the property. $n = b+a$ cannot be a consequence of $A(n)$ since $b+a$ does not belong to the language $L$ of $A(x)$. If

$$S_1 \models \forall x x = z$$

then $S_1 \models n = 0$ and $S_1 \models n = 1$, and hence $S_1 \models 0 = 1$. This is impossible since $S_1$ is consistent. If

$$S_1 \models \forall y \forall z n = y+z$$

then $S_1 \models n = 0$ and $S_1 \models n = 1$, and hence $S_1 \models 0 = 1$. This is impossible since $S_1$ is consistent.
then \( S_1 = n = 0+0 \) and \( S_2 = n = 0+1 \), and again \( S_3 = 0 = 1 \). Thus \( S_1 \), and hence \( S \), must be \( s \)-consistent.

We see that \( S \) is a counterexample to Conjecture 2.10 (II). We now modify this conjecture. First we introduce a generalisation of the notion of \( s \)-consistency.

2.13 DEFINITION. Let \( T \) be a theory in \( L \).
(i) \( T \) is \( s(n) \)-consistent \( \iff \) there is no formula \( A(x_1, \ldots, x_n) \) of \( L \) with only \( x_1, \ldots, x_n \) free such that
\[
T \vdash A(k_1, \ldots, k_n) \quad \text{for all } k_1, \ldots, k_n \in \mathbb{N}
\]
(ii) \( T \) is \( s(\infty) \)-consistent \( \iff \) \( T \) is \( s(n) \)-consistent for all \( n \in \mathbb{N} \).

2.14 LEMMA. (I) \( s(0) \)-consistency is ordinary consistency, \( s(1) \)-consistency is \( s \)-consistency as defined in Definition 1.2.
(II) \( T \) is \( s(n+1) \)-consistent \( \Rightarrow \) \( T \) is \( s(n) \)-consistent.
(III) For every \( n \), there are theories \( T \supseteq S_0 \) such that \( T \) is \( s(n) \)-consistent but not \( s(n+1) \)-consistent.
(IV) \( \mathcal{N} \models T \Rightarrow T \) is \( s(\infty) \)-consistent.

PROOF:
(III) By Theorem 1.7, \( T = PA + \neg \forall x \, G(x) \) is \( s(0) \)-consistent and \( s(1) \)-inconsistent. The theory
\[
T = S_0 + (\exists x \, \exists y \, x + y = y + x)
\]
of \( \mathbb{Z} \) is \( s(1) \)-consistent and \( s(2) \)-inconsistent. Similar examples can be produced for any \( n \geq 1 \).

2.15 THEOREM. Let \( T \supseteq PA \). Then for \( n \geq 1 \),
(i) \( T \) is \( s(n) \)-consistent \( \iff \) \( T \) is \( s(1) \)-consistent.
(ii) \( T \) is \( s(\infty) \)-consistent \( \iff \) \( T \) is \( s(1) \)-consistent.

PROOF:
(i) \( \Rightarrow \): Use the lemma.
(ii) \( \Rightarrow \): Assume \( T \) is \( s(n) \)-inconsistent for some \( n > 1 \). Then
\[
T \vdash A(k_1, \ldots, k_n) \quad \text{for all } k_1, \ldots, k_n
\]
(2-9) \( T \vdash \exists x_1 \ldots \exists x_n \neg A(x_1, \ldots, x_n) \)
(2-10) \( T \vdash \exists x_1 \ldots \exists x_n A(x_1, \ldots, x_n) \)
There is a recursive injection
\[
f_n : \mathbb{N}^n \rightarrow \mathbb{N}
\]
\[
f_n(x_1, \ldots, x_n) = 2^{x_1+1} \cdot 3^{x_2+1} \cdot \ldots \cdot p_{x_n+1}^{x_n+1}
\]
where \( p_i \) is the \( i \)-th prime number. There is also a recursive bijection
\[
f_n : \mathbb{N}^n \rightarrow \mathbb{N}
\]

Define
\[
\Phi_n : \mathbb{N}^n \rightarrow \mathbb{N}
\]
\[
\Phi_n = f_n \circ \Phi
\]
Then \( \Phi_n \) is a recursive bijection of \( \mathbb{N}^n \) and \( \mathbb{N} \). Since \( \Phi_n \) is recursive, it is definable in \( T \). Define in \( T \)
\[
B(x) \leftrightarrow \Phi_n^{-1}(x)
\]
Then
\[
T \vdash A(x_1, \ldots, x_n) \leftrightarrow B(\Phi_n(x_1, \ldots, x_n))
\]
From (2-9) and (2-10), respectively, we infer
(2-11) \( T \vdash B(n) \) for all \( n \)
(2-12) \( T \vdash \exists x_1 \ldots \exists x_n \neg B(\Phi_n(x_1, \ldots, x_n)) \)
From (2-12), by predicate logic, we get
(2-13) \( T \vdash \exists x \neg B(x) \)
(2-11) and (2-13) imply that \( T \) is \( s(1) \)-inconsistent.
(II) is an immediate corollary to (I).

2.16 CONJECTURE. We now modify Conjecture 2.16 (II).
(II*) Assume \( S_0 \supseteq S \). Then \( S \) is \( s(\infty) \)-consistent \( \iff \) \( \mathcal{N} \models S \).
It is an open problem whether (II*) is true.

2.17 LEMMA. Conjecture 2.16 (II*) implies Conjecture 2.16 (III).

PROOF:
Assume \( S \supseteq PA \) and that \( S \) is \( s \)-consistent. By Theorem 2.15, \( S \) is \( s(\infty) \)-consistent. By Conjecture 2.16 (II*), it follows that \( \mathcal{N} \models S \).

2.18 On Conjecture 2.10 (III). Conjecture 2.10 (III) is still an open problem. The only known first-order elementary extensions of PA are obtained by adding Gödel sentences or negations of Gödel sentences
so PA. All such known extensions are either α-consistent and true in \( \mathcal{N} \) or α-inconsistent and false in \( \mathcal{N} \). Thus nothing is known which precludes that the conjecture is true.

The problem of the truth of Conjecture 2.10 (III) is of no doubt non-trivial. It has proved very difficult to find extensions of PA other than those obtained by adding Gödel sentences or their negations. Therefore it will be difficult to find a counterexample to the conjecture. On the other hand, the equivalence

\[ S \text{ is } \alpha\text{-consistent } \iff \mathcal{N} \models S \]

appears to be so remarkable that a trivial proof of it seems unlikely.

One possible proof procedure may be based on Lemma 2.17. The problem of proving Conjecture 2.16 (II*) may be more tractable than a direct proof of Conjecture 2.10 (III). Another strategy to try might be to show that every extension of PA can be obtained by adding Gödel sentences or negations of Gödel sentences. Then the desired conclusion \( \mathcal{N} \models S \) would follow by Lemma 2.9.

2.19 CONJECTURE. For \( \alpha(0) \)-consistency (ordinary consistency), we have

If \( T \) is consistent, then \( T \) has a complete and consistent extension.

We may conjecture the following weak analogue for \( \alpha(1) \)-consistency (i.e., \( \alpha \)-consistency).

Conjecture IV. Let \( T \) be a theory in \( L \) such that \( S_0 \subseteq T \) and \( T \) is \( \alpha \)-consistent. Then \( T \) has an \( \alpha \)-complete and \( \alpha \)-consistent extension.

This conjecture is false. We note first that by Theorem 2.4, \( T \) can have only one \( \alpha \)-complete extension, \( \text{Th}(\mathcal{N}) \). For a counterexample, it therefore suffices to find an \( \alpha \)-consistent extension \( T \) of \( S_0 \) such that \( \mathcal{N} \models T \). Such a theory \( T \) was found in § 2.12. Let

\[ T = S_0 + [\exists x \exists y x + y \neq y + x] \]

2.20 OPEN PROBLEMS. (I) Prove or disprove Conjecture 2.16 (II*). If Conjecture (II*) can be disproved, find further conditions on \( S \) which together with \( \alpha(\omega) \)-consistency implies that \( S \) is true in \( \mathcal{N} \).

(II) Prove or disprove Conjecture 2.10 (III). If Conjecture 2.10 (III) can be disproved, find further conditions on \( S \) which together with \( \alpha(\omega) \)-consistency implies that \( S \) is true in \( \mathcal{N} \).

(III) For ordinary consistency, \( \alpha(0) \)-consistency, we have the following model theoretic criterion.

\( T \) is \( \alpha(0) \)-consistent \( \iff \) \( T \) has a model.

Can we find and prove similar simple model theoretic criteria of \( \alpha \)-consistency? How \( \alpha(\omega) \)-consistency for \( \alpha > 1 \) of \( \alpha(\omega) \)-consistency? If Conjecture 2.10 (III) is true, it will give a partial answer to this question:

Let \( T \) be a theory such that \( \text{PA} \subseteq T \). Then \( T \) is \( \alpha(1) \)-consistent \( \iff \) \( T \) is true in the standard model.

2.21 REMARK. Of course, there are simpler examples of theories which are consistent and \( \alpha \)-inconsistent than those we considered in the proof of Theorem 1.7. Let, e.g.,

\[ T = S_0 + [\exists x \exists y x = y] \]

Then \( T \) is certainly \( \alpha \)-inconsistent since

\[ T \models 0 \neq 0, T \models 1 \neq 0, \ldots, T \models \exists x x = x \]

We define a model \( M = (\mathbb{M}, 0, '+', '\cdot', '\cdot') \) by letting

\[ \mathbb{M} = \{0, 1, 2, \ldots, \omega\} \]

0 and the functions '+', '\cdot', '\cdot' are defined in the usual way on the natural numbers in \( \mathbb{M} \). We define for all \( n \in \mathbb{M} \),

\[ \omega' = a + \omega = a \omega + a = a \cdot \omega = a \cdot a = \omega + \omega \]

Then

\[ \mathcal{M} = S_0 \]

and also

\[ \mathcal{N} \models [\exists x x = x'] \]

since \( \mathcal{N} \models \omega = \omega' \). Now \( T \) has a model and therefore is consistent.

Such theories as \( T \) have, however, very limited consequences for the question of arithmetical truth since they occur before the axiomatic development of arithmetic has reached a level where Gödel's theorem can be proved.
8. Proofs and Examples in Logic and Geometry

REMARK. Working with the theorems and problems in books on logic and mathematics, I have occasionally found proofs and examples which appear to be new or contain some new details. Some of these proofs and examples are collected here. It is unlikely that they all contain something new; but they are certainly the results of my own independent work. I have tried to include such proofs and examples which seem to be at least sufficiently interesting to appear as an example or exercise in a textbook.

Suitable background is a good course in mathematical logic, e.g., based on Shoenfield (1967) or Bell and Machover (1977). Some knowledge of Hilbert space theory and classical geometry will also be helpful.

8-1 Padoa's Method: An Example

1.1 DEFINITION. (I) Let $T$ be a first-order theory with language $L$. If $U$ is a nonlogical symbol in $L$, then we denote by $L_{(U)}$ the language $L_{(U)} = L - \{U\}$.

(II) Let $P$ be an $n$-place predicate of $L$. A definition in $T$ of $P$ in terms of $L_{(P)}$ is a theorem of $T$

$$T = \forall x_1 \ldots \forall x_n (P(x_1, \ldots, x_n) \leftrightarrow A(x_1, \ldots, x_n))$$

where $A(x_1, \ldots, x_n)$ is a formula of $L_{(P)}$ having only $x_1, \ldots, x_n$ free.

(III) Let $f$ be an $n$-place function symbol of $L$. A definition in $T$ of $f$ in terms of $L_{(P)}$ is a theorem of $T$
\[ T \models \forall x_1 \ldots \forall x_n \forall y (y = s(x_1, \ldots, x_n) \leftrightarrow A(x_1, \ldots, x_n, y)) \]
where \( A(x_1, \ldots, x_n, y) \) is a formula of \( L(y) \) having only \( x_1, \ldots, x_n, y \) free and satisfying the existence and uniqueness conditions
\[ T \models \forall x_1 \ldots \forall x_n \exists y A(x_1, \ldots, x_n, y) \]
\[ T \models \forall x_1 \ldots \forall x_n \forall y \forall z (A(x_1, \ldots, x_n, y) \land A(x_1, \ldots, x_n, z) \rightarrow y = z) \]
when \( z \) is free for \( y \) in \( A(x_1, \ldots, x_n, y) \).

(IIV) Let \( c \) be a constant in \( L \). A definition in \( T \) of \( c \) in terms of \( L(c) \) is a theorem of \( T \)
\[ T \models \forall y (c = y \leftrightarrow A(y)) \]
where \( A(y) \) is a formula of \( L(c) \) having only \( y \) free and satisfying
\[ T \models \exists y A(y) \]
\[ T \models \forall y \forall z (A(y) \land A(z) \rightarrow y = z) \]
when \( z \) is free for \( y \) in \( A(y) \).

1.2 DEFINITION. Let \( U \) be a symbol of \( L \). \( U \) is independent in \( T \) if there is no definition in \( T \) of \( U \) in terms of \( L(U) \).

1.3 Padova's Method. Padova's method gives a way of demonstrating that a symbol \( U \) is independent in \( T \): Find two models \( M_1 \) and \( M_2 \) of \( T \) with the same domain \( M = M_1 = M_3 \) such that any symbol of \( L(U) \) gets the same interpretation in \( M_1 \) and \( M_2 \) while \( U \) gets different interpretations in \( M_1 \) and \( M_2 \).

Let, e.g., \( U \) be the predicate \( P \) and assume that there are two models \( M_1 \) and \( M_2 \) of \( T \) satisfying the conditions in Padova's method. Suppose \( P \) is definable in \( T \) in terms of \( L(P) \). Then for all individuals \( a_1, \ldots, a_n \in M_1 \),
\[ M_1 \models P(a_1, \ldots, a_n) \leftrightarrow M_2 \models A(a_1, \ldots, a_n) \]
\[ M_2 \models A(a_1, \ldots, a_n) \leftrightarrow M_2 \models P(a_1, \ldots, a_n) \]
The first and third equivalence follow by Definition 1.1. The second equivalence follows since \( M_1 \) and \( M_2 \) agree on \( L(P) \). Thus \( M_1 \) and \( M_2 \) give the same interpretation to \( P \), contrary to the choice of \( M_1 \) and \( M_2 \). Therefore \( P \) is not definable in terms of \( L(P) \).

In this section, we give a simple example of an application of Padova's method.

1.4 EXAMPLE. Let \( L = \{ R, P_1, P_2, \ldots \} \) where \( R \) is a binary predicate, \( P_1, P_2, \ldots \) are unary predicates.

Let \( T \) be the theory such that \( L(T) = L \) and \( T \) has as axioms all the sentences
\[ A_{ij}: \forall x \forall y (R(x, y) \rightarrow P_i(x) \land P_j(y)) \]
for \( i, j \in \mathbb{N} \). Then \( R \) is independent of \( P_1, P_2, \ldots \) as can be shown by Padova's method. We may, e.g., choose
\[ M = M_1 = M_2 = \{ a \} \]
\[ P_k = \{ a \} \text{ in } M_1 \text{ and } M_2 \text{ for } k \in \mathbb{N} \]
\[ R = \emptyset \text{ in } M_1 \]
\[ R = \{ (a, a) \} = M \times M \text{ in } M_2 \]

Clearly,
\[ M_1 \models T \]
\[ M_2 \models T \]

Any \( P_k \) has the same interpretation in \( M_1 \) and \( M_2 \) while \( R \) has different interpretations in \( M_1 \) and \( M_2 \). It is therefore impossible to define \( R \) in terms of \( P_1, P_2, \ldots \) in \( T \).

1.5 EXAMPLE. Let \( L(T_2) = L \) as in Example 1.4, and let \( T_2 \) have the axioms
\[ B_{ij}: \forall x \forall y (P_i(x) \land P_j(y) \rightarrow R(x, y)) \]
for \( i, j \in \mathbb{N} \). Then in \( T_2 \) \( P \) is independent of \( P_1, P_2, \ldots \) as can be shown by Padova's method.

1.6 EXAMPLE. Let \( L(T_3) = L \) and let \( T_3 \) as axioms have all the sentences \( A_{ij} \) and all the sentences \( B_{ij} \). Now R is not independent of \( P_1, P_2, \ldots \) in \( T_3 \). Indeed, it suffices to choose one \( i \) and one \( j \) and the two sentences \( A_{ij} \) and \( B_{ij} \) to get a defining theorem.
8-2 Decidability

2.1 DEFINITION. The function \( F : \mathbb{N} \rightarrow \mathbb{N} \) enumerates the set \( A \) iff \( A = \{ F(n) \mid n \in \mathbb{N} \} \).

2.2 LEMMA. Let \( A \) be an infinite recursively enumerable set (an RE set). Then there is an injective recursive function which enumerates \( A \).

2.3 LEMMA. Let \( A \) be an infinite set. Then \( A \) is recursive iff there is a strictly increasing recursive function \( F \) which enumerates \( A \).

2.4 LEMMA. If \( T \) is an axiomatised and complete theory, then \( T \) is decidable.

2.5 THEOREM (Lov-Vaught). Let \( c \) be an infinite cardinal and \( T \) a consistent theory such that \( \text{cd}(L(T)) \leq c \), \( T \) has only infinite models, and \( T \) is categorical in power \( c \). Then \( T \) is complete.

2.6 DEFINITION. We use \( E_n \) (sometimes written \( E(x) \)) to denote a sentence in \( L = \emptyset \) expressing that there are exactly \( n \) individuals. Thus, e.g., \( E_2 \) may be

\[
E_2 \equiv \exists x \exists y (x \neq y \land \forall z (z = x \lor z = y))
\]

2.7 LEMMA. Let \( T \) be a finitely axiomatised theory with \( \text{L}(T) \) finite. Then there is a decision method which for all formulas of \( \text{L}(T) \) of the form \( E_n \rightarrow \Delta \) decides whether \( \Delta \) is a theorem in \( T \).

2.8 LEMMA. Let \( L \) be any first-order language, \( A \) a formula of \( L \), and \( \kappa \in \mathbb{N} \). Then for any model \( \mathcal{M} \) for \( L \) with \( \text{cd}(\mathcal{M}) = \kappa \), it is decidable whether \( \mathcal{M} \models A \) or \( \mathcal{M} \not\models A \).

2.9 THEOREM. Let \( L(T) \) be finite. Assume that \( T \) is finitely axiomatised and categorical in power \( c \) for some infinite cardinal \( c \). Then \( T \) is decidable.

PROOF:
Let \( \Gamma = \{ \neg E_1, \ldots, \neg E_n \} \) and \( T^* = T + \Gamma \), i.e., \( T^* \) is obtained from \( T \) by adding all the \( \neg E_i \) as new nonlogical axioms. Since \( \Gamma \) is decidable, \( T^* \) is axiomatised. There are three possible alternatives for a theory \( \Gamma \) satisfying the hypothesis of the theorem: (1) \( \Gamma \) is inconsistent; (2) \( \Gamma \) is consistent but \( T^* \) is inconsistent; (3) \( T^* \) is consistent. We consider each of these three cases.

(1) \( \Gamma \) is inconsistent: Then every sentence of \( L(T) \) is a theorem of \( \Gamma \). Since \( L(T) \) is decidable, so is \( \Gamma \).

(2) \( \Gamma \) is consistent and \( T^* \) is inconsistent: Then in turn,

\[
(2-1) \quad T^* \vdash \bot
\]

\[
(2-2) \quad T \vdash \neg E_1 \land \cdots \land \neg E_n \rightarrow \bot
\]

\[
(2-3) \quad T \vdash E_1 \lor \cdots \lor E_n
\]

for some \( n \). Let \( m \) be the smallest Gödel number of a proof in \( T^* \) of \( \bot \). Since \( T^* \) is axiomatised, \( m \) can be found recursively. Let \( I = (\neg E(n_1), \ldots, \neg E(n_k)) \) be the sequence of all the axioms in \( \Gamma \) occurring in the proof with Gödel number \( m \) and ordered such that \( n_1 < \cdots < n_k \). Then \( I \) can be computed effectively. Clearly \( k > 0 \) since otherwise \( \Gamma \) is inconsistent. Let \( n = n_k \). It follows that every model of \( T \) has at most \( n \) elements.

Consider any \( k, 1 \leq k \leq n \). Since \( L(T) \) is finite, there are only finitely many nonisomorphic models with \( k \) elements: \( M_{k,1}, \ldots, M_{k,p(k)} \). Once \( k \) is given, these models can be effectively constructed and \( p(k) \) be computed. The total number of nonisomorphic models for \( L(T) \) is the computable number
\[ \alpha = p(1) + p(2) + \ldots + p(n) \]

Let \( A_1, \ldots, A_q \) be all the axioms of \( T \) and let \( B \) be any sentence in \( L(T) \). Then

\[ (2-4) \quad T \vdash B \iff \vdash A_1 \land \ldots \land A_q \rightarrow B \]

By Lemma 2.8, there is a decision method for the last statement in (2-4).

(3) \( T^* \) is consistent: Then \( T^* \) has only infinite models and \( T \) and \( T^* \) have the same infinite models. Therefore \( T^* \) is categorical in power \( c \).

By Los-Vaught’s theorem, \( T^* \) is complete and therefore decidable by Lemma 2.4.

Let \( B \) be any sentence of \( L(T) \). Apply the decision method of \( T^* \) to \( B \). If the answer is \( T^* \vdash B \), then \( T \vdash B \) since \( T^* \) is an extension of \( T \). Now assume that the answer is \( T^* \nvdash B \). Since only finitely many axioms of \( T^* \) can be used in a proof of \( B \), there is a \( n \) such that

\[ (2-5) \quad T \vdash \neg E_{i_1} \land \ldots \land \neg E_{i_n} \rightarrow B \]

which by sentential logic is equivalent with

\[ (2-6) \quad T \vdash E_1 \lor \ldots \lor E_n \lor B \]

First we must calculate an \( n \) for which (2-6) is satisfied. This is done by the same method as for Proposition (2-3). Now consider all the sentences \( E_k \rightarrow B \), \( k = 1, \ldots, n \). By Lemma 2.7, \( T \vdash E_k \rightarrow B \) is decidable.

(2-7) \[ \text{If } T \vdash E_k \rightarrow B \text{ for } k = 1, \ldots, n, \text{ then } T \vdash B \text{ by (2-6) and sentential logic.} \]

(2-8) \[ \text{If } T \nvdash E_k \rightarrow B \text{ for some } k, \text{ then } T \nvdash B \]

since \( T \vdash B \) implies \( T \vdash E_k \rightarrow B \) for all \( k \) by sentential logic. Thus the decidability of \( T \vdash E_k \rightarrow B \) together with (2-7) and (2-8) gives a decision method for \( T \vdash B \).

2.10 REMARK. The theorem shows that for every \( T \) satisfying the hypothesis, there is some decision method associated with \( T \). A peculiarity about the theorem is that this association is nonrecursive. As soon as we know whether \( T \) is inconsistent, or \( T \) is consistent and \( T^* \) is inconsistent, or \( T^* \) is consistent, then we can effectively associate a decision method with \( T \) as in the proof of the theorem. The trouble is that given a theory \( T \) of which we only know that \( T \) satisfies the hypothesis of the theorem, there is no algorithm which decides whether \( T \) is inconsistent, or \( T \) is consistent and \( T^* \) is inconsistent, or \( T^* \) is consistent.

2.11 THEOREM. There is an axiomatised theory \( T \) such that \( L(T) \) is finite, \( T \) is categorical in some infinite power \( c \), and \( T \) is undecidable.

PROOF:

Let \( L = L(T) = \{ \} \). Let \( A \subset \mathbb{N} \) be RE but not recursive. Define \( T_A = \{ \neg E_n \mid n \in A \} \). Let \( T^*_A \) be the theory in \( L \) having \( \Gamma_A \) as its set of nonlogical axioms. Since \( A \) is nonrecursive, it is infinite. By Lemma 2.2, there is a recursive bijection \( F: \mathbb{N} \rightarrow A \) which enumerates \( A \), i.e.,

\[ (2-9) \quad A = \{ F(n) \mid n \in \mathbb{N} \} \]

Define for \( n = 0, 1, 2, \ldots \)

\[ C_n = \neg E_{F(0)} \land \ldots \land \neg E_{F(n)} \]

Let \( T \) be the theory in \( L \) whose nonlogical axioms consist of all the \( C_n \). Then \( T \) and \( T^*_A \) are equivalent. Define

\[ G: \mathbb{N} \rightarrow \mathbb{N} \]

\[ G(n) = \text{if } n = 0 \text{ then } 0 \text{ otherwise } F(n) \]

where \( U^* \) is the Gödel number of the expression \( U \). Then \( G \) enumerates \( A \subset \mathbb{Q} \). Define \( \alpha(n) = G(n) + G(n+1) \) for all \( n \). By Lemma 2.3, \( \alpha \) is recursive so that \( T \) is axiomatised and therefore \( T^*_A \) axiomatisable.

Since \( L(T) = \{ \} \), every model \( M \) for \( L(T) \) consists only of the domain \( M \) without any structure induced. Therefore any two models of \( T \) of the same cardinality are isomorphic. It follows that \( T \) is categorical in all infinite powers.

Suppose that \( T \) is decidable. Then \( Thm_T \) is recursive. We show

\[ (2-10) \quad n \in A \iff T \vdash \neg E_n \]

If \( n \in A \), then \( \neg E_n \) is an axiom of \( T^*_A \) and hence a theorem of \( T \). If \( n \notin A \), then there is a model of \( T^*_A \) and therefore of \( T \), having exactly \( n \) elements. By the completeness theorem, \( T \vdash \neg E_n \).

Define \( H : \mathbb{N} \rightarrow \mathbb{N} \) by \( H(n) = \neg E_n \). Then \( H \) is recursive. By (2-10),

\[ (2-11) \quad A(n) \iff Thm_T(\neg E_n) \iff Thm_T(H(n)) \]
It follows that \( A \) is recursive, contrary to the choice of \( A \). Therefore \( T \) cannot be decidable.

2.12 REMARK. Theorems 2.9 and 2.11 occur as problems in Shoenfield (1967). The possible novelty in the proof of Theorem 2.11 is the use of lemmas 2.2 and 2.3 which apparently simplifies the proof somewhat.

8-3 Recursion Theory

3.1 REMARK. For the theorems in this section, we need recursion theory at a level where it will be too space consuming to give and explain all concepts, notation, and results needed as preliminaries. The reader may consult Shoenfield (1967), Chapter 7, or Rogers (1967).

Small Greek letters \( \alpha, \beta, \ldots \) represent total unary functions. Small Latin letters \( a, b, \ldots, x, y, z \) represent numbers. Each capital written letter \( \mathcal{A}, \mathcal{B}, \ldots \) represents a sequence of Greek letters followed by a sequence of Latin letters. E.g., \( \mathcal{A} \) may be the \( (m+n) \)-sequence \( \alpha_1, \ldots, \alpha_m, a_1, \ldots, a_n \). In that case, let \( \mathcal{A}(x) \) represent the sequence \( \alpha_1(x), \ldots, \alpha_m(x), a_1, \ldots, a_n \). A sequence of Latin letters \( a_1, \ldots, a_n \) may be represented by a bold letter like \( a \), or \( b \), \ldots A set which is designated by an indexed letter like \( A_{i,k} \) may also be denoted by \( A(i,k) \) if this is more convenient for typographical reasons. Similarly, if a number variable \( n_i \) is indexed by \( i \), we may write \( a(i) \) instead. This situation occurs when \( n_i \) is index to another letter, e.g., \( A_n(x) \).

Every \((m,n)\)-ary recursive partial functional \( F \) has an index \( f \), its Gödel number. Conversely, given any number \( k \in \mathbb{N} \), there is a unique \((m,n)\)-ary recursive partial functional having \( k \) as an index. This functional is designated by \( \langle k \rangle_{m,n} \). The superscripts may be omitted when it is clear from the context that \( \langle k \rangle \) denotes an \( (m,n)\)-ary partial functional.

3.2 DEFINITION. Let \( Q \) be a \((1,1)\)-ary relation such that for all sets \( A \) and \( B \),

\[\begin{align*}
(3-1) & \quad A \subseteq B \Rightarrow \forall x (Q(K_A, x) \rightarrow Q(K_B, x))
\end{align*}\]

A set \( A \) is \( Q \)-closed if \( \forall x (Q(K_A, x) \rightarrow A(x)) \). Let \( A_Q \), sometimes written as \( A(Q) \), denote the intersection over the class of all the \( Q \)-closed sets.

3.3 LEMMA. \( A_Q \) is \( Q \)-closed.

PROOF:
Assume \( Q(K_A, x) \). We want to show \( A_Q(x) \). Let \( B \) be any \( Q \)-closed set. Then \( A \subseteq B \). By \( (3-1) \), \( \forall x (Q(K_A, x) \rightarrow Q(K_B, x)) \). Then \( B(x) \) since \( B \) is \( Q \)-closed. Since \( x \) is in every \( Q \)-closed set, \( x \) is in their intersection \( A_Q \).

3.4 LEMMA. \( A_Q(x) \iff \forall x (\forall \alpha(x) \leq 1 \Rightarrow (\forall x (Q(\alpha, x) \rightarrow \alpha(x) = 0) \rightarrow \alpha(x) = 0)) \)

PROOF:

\( \Rightarrow \) Choose \( \alpha \) and \( \alpha(x) \)

\[\begin{align*}
(3-2) & \quad A_Q(x) \\
(3-3) & \quad \forall x \alpha(x) \leq 1 \\
(3-4) & \quad \forall x (Q(\alpha(x) \rightarrow \alpha(x) = 0))
\end{align*}\]

Since for every \( x \), \( \alpha(x) \in \{0, 1\} \), \( \alpha \) is the characteristic function of some set \( B \), \( \alpha = K_B \). Applying this to \( (3-4) \) gives

\[\begin{align*}
\forall x (Q(K_B, x) \rightarrow B(x))
\end{align*}\]

and \( B \) is \( Q \)-closed. Since \( A_Q \subset B \), \( B(x) \). Therefore \( \alpha(x) = K_B(x) = 0 \).

\( \Leftarrow \Rightarrow \) Choose \( \alpha = K_A \).

3.5 EXAMPLE. Shoenfield gives Lemma 3.4 in the form

\[\begin{align*}
(3-5) & \quad A_Q(x) \quad \forall \alpha (\forall x (Q(\alpha, x) \rightarrow \alpha(x) = 0) \rightarrow \alpha(x) = 0))
\end{align*}\]

But this cannot be proved as the following counterexample shows.

Let \( P \) be the set of prime numbers. Define \( Q \) by

\[\begin{align*}
(3-6) & \quad Q(\alpha(x) \rightarrow \exists \beta (\beta = K_B \land P(\beta))
\end{align*}\]

We prove

\[\begin{align*}
(3-7) & \quad C \text{ is } Q \text{-closed} \Rightarrow P \subseteq C
\end{align*}\]

Assume \( C \) is a \( Q \)-closed set and \( n \in P \). Then

\[\begin{align*}
\forall x (Q(K_{x}, x) \rightarrow C(x))
\end{align*}\]
Hence

(3.8) \( Q(K_C, n) \rightarrow C(n) \)

Since \( n \in P \), \( K_C = K_C \land P(n) \). By (3.6), \( Q(K_C, n) \). Then by (3.8), \( n \in C \).

\( Q(K_p, x) \) implies \( P(x) \) by (3.6). Therefore \( P \) is \( Q \)-closed. This together with (3.7) implies

(3.9) \( \alpha_Q = P \)

Now suppose that (3.5) is true. Let \( a \) be a prime number. By (3.9) and (3.5),

(3.10) \( \forall \alpha \left[ \forall x \left( Q(\alpha, x) \rightarrow \alpha(x) = 0 \right) \rightarrow \alpha(x) = 0 \right] \)

Define \( \alpha \) by

\( \alpha(x) = 2 \)

Since \( \alpha \) is not the characteristic function of any set, we have

\( \forall x \neg Q(\alpha, x) \). Therefore

\( \forall x \left( Q(\alpha, x) \rightarrow \alpha(x) = 0 \right) \)

By (3.10), \( \alpha(n) = 0 \) which contradicts the definition of \( \alpha \).

3.6 DEFINITION. An \( H \)-index is a sequence number satisfying the following:

H1. If \( e \) is a number, then \(< 0, e> \) is an \( H \)-index.

H2. If \( e \) is an \( H \)-index, then \(< 1, e> \) is an \( H \)-index.

H3. If \( W_{e,0,1} \) is a set of \( H \)-indices, then \(< 2, e> \) is an \( H \)-index.

3.7 THEOREM (Kleene's Recursion Theorem). Let \( G \) be a recursive \((m, n+1)\)-ary partial functional. Then there is a recursive \((m, n)\)-ary partial functional \( F \) with index \( f \) such that \( F(n) = G(A, f) \) for all \( A \).

3.8 THEOREM. For any \( H \)-index \( i \), let the set \( A_i \) be defined by

\[
A_i(x) \iff \begin{cases} 
  x = i & \text{if } i_0 = 0 \\
  x = i \lor A_{i+1}(x) & \text{if } i_0 = 1 \\
  x = i \lor \exists y \left( y \in W_{i-1} \land A_y(x) \right) & \text{if } i_0 = 2 
\end{cases}
\]

Then there is a recursive function \( F \) such that \( A_i = W_{F(i)} \) for every \( H \)-index \( i \).

PROOF:
Let \( F \) be a recursive function. We now construct the skeleton of a proof of the identity \( A_i = W_{F(i)} \). Noting what further properties of \( F \) are sufficient to give a complete proof. At the end, we show that there is indeed a recursive function \( F \) having these properties. The proof is by induction on the definition of \( H \)-indices.

Let \( i = <0, e> \). Then

\( A_i(x) \iff i = x \)

Let \( k \) be an \( RE \)-index of \( e \). Then

\( A_j(x) \iff W_k(l_x) \iff W_{S(j, 2)}(x) \)

Therefore, in this case, it suffices to have

(3.11) \( P(i) = S_{0,1,1}(k, l) \)

Let \( i = <1, e> \). Then

\( A_i(x) \iff x = i \lor A_{i+1}(x) \)

Define \( P \) by

\( P(x, i, a) \iff x = i \lor \exists y \left( T_1(a, x, y) \iff W_{p}(x, i, a) \right) \)

where \( p \) is an \( RE \)-index of \( P \). Then

\( A_j(x) \iff x = i \lor A_{i+1}(x) \)

\( \iff x = i \lor W_{F(i)}(x) \)

\( \iff x = i \lor \exists y \left( T_1(l_x, y) \iff W_{p}(x, i, y) \right) \)

\( \iff W_{p}(x, i, F(i)) \)

\( \iff W_{S(p, i, F(i))}(x) \)

In this case, it suffices to have

(3.12) \( P(i) = S_{0,1,2}(p, i, F(i)) \)

Let \( i = <2, e> \). Then

\( A_i(x) \iff x = i \lor \exists y \left( W_{i+1}(y) \land A_y(x) \right) \)

\( \iff x = i \lor \exists y \left( W_{i+1}(y) \land W_{F(y)}(x) \right) \)

(* by the induction hypothesis *)

\( \iff x = i \lor \exists y \left( T_1(l_x, y, z) \iff T_1(F(y), x, z) \right) \)
Define Q by
\[ Q(x, y, a) \iff x = 1 \lor \exists y (\exists z T_1(x, y, a) \land \exists z T_1(f(y), x, x)) \]
Then Q is RE. Let \( q \) be an RE-index of Q. Then, whenever \( F \) is the recursive partial function \( f \),
\[ A_q(x) \iff Q(x, f(x), q, f(x)) \]
\[ \iff W_q(x, f(x), q, f(x)) \]
\[ \iff W_{S(q, f(x), q, f(x))}(x) \]
In this case, it therefore suffices to have
\[ (3-13) \quad F(i) \equiv S_{q, f(x), q, f(x)}(x) \]
where \( f \) is an index of \( F \).
By the recursion theorem, there is a recursive partial function \( F \) satisfying (3–11), (3–12), and (3–13). Since \( S_{m, n, k} \) is total, so is \( F \).

**8-4 Arithmetic Theories**

4.1 DEFINITION. \( T' \) is an extension of \( T \) if \( L(T) \subseteq L(T') \) and every theorem of \( T \) is a theorem of \( T' \).

4.2 DEFINITION. Peano arithmetic PA is the theory with language \( L(PA) = \{ 0, S, +, \cdot, < \} \) and the nonlogical axioms

\begin{align*}
PA_1 & : S(x) \neq 0 \\
PA_2 & : S(x) = S(y) \rightarrow x = y \\
PA_3 & : x + 0 = x \\
PA_4 & : x + S(y) = S(x + y) \\
PA_5 & : x \cdot 0 = 0 \\
PA_6 & : x \cdot S(y) = x \cdot y + x \\
PA_7 & : \neg(x < 0) \\
PA_8 & : x < S(y) \leftrightarrow x < y \lor x = y \\
PA_9 & : A(0) \land \forall x (A(x) \rightarrow A(S(x))) \rightarrow A(x)
\end{align*}

for any formula \( A(x) \) of \( L(PA) \).

4.3 DEFINITION. By an arithmetic theory we here mean any extension of PA.

4.4 DEFINITION. A PA-theory is an extension \( T \) of PA such that for every formula \( A(x) \) of \( L(T) \),
\[ T \vdash A(0) \land \forall x (A(x) \rightarrow A(S(x))) \rightarrow A(x) \]

4.5 DEFINITION. A theory \( T \) is open if all the nonlogical axioms of \( T \) are open, i.e., they are quantifier free.

4.6 LEMMA. Let \( T \) be a PA-theory and \( T' \) an extension by definitions of \( T \). Then \( T' \) is a PA-theory.

4.7 THEOREM. Assume that there is a finitely axiomatised consistent PA-theory. Then there is a finitely axiomatised consistent open PA-theory.

PROOF:
Let \( T \) be a finitely axiomatised theory all of whose nonlogical axioms are in prenex normal form. By the quantifier index of \( T \) we mean an ordered pair \( (l, m) \) where \( l \) is the maximal length of a prefix of a nonlogical axiom in \( T \) and \( m \) is the number of axioms having a prefix of length \( l \). We order the indices by

\[ (l, m) < (l', m') \iff l < l' \lor (l = l' \land m < m') \]

If \( (l, m) = (0, m) \), then \( T \) is open.

Let \( T \) be a theory satisfying the hypothesis of the theorem. By the theorem on prenex form, we may assume that all the axioms of \( T \) are in PNF. We prove the theorem by induction on the quantifier index \( (l, m) \) of \( T \).

If \( l = 0 \), then \( T \) is open and we are at our goal. Assume \( l > 0 \). Let \( A_{11}, \ldots, A_{mm} \) be the axioms of length \( l \). Consider \( A_1 \). We get two cases:
1. \( A_1 \) is of the form \( \forall x B(x) \), and
2. \( A_1 \) is of the form \( \exists x B(x) \).

\( A_1 \) is \( \forall x B(x) \): Obtain \( T' \) from \( T \) by replacing \( A_1 \) by \( B(x) \). Then \( T' \) and \( T \) are equivalent and therefore \( T' \) is a PA-theory. Let \( (l', m') \) be the quantifier index of \( T' \). If \( m = 1 \), then \( l' = l \) and hence \( (l', m') < (l, m) \). If \( m > 1 \), then \( l' = l \) and \( m' < m \) and again \( (l', m') < (l, m) \).
$A_f$ is $\exists x B(x)$: Obtain $T_1$ from $T$ by letting $L(T_1) = L(T) \cup \{f\}$ where $f$ is a new function symbol with the following defining axiom, which is also added to $T$:

\begin{equation}
B(f(y_1, \ldots, y_n)) \land \forall x \left( x < f(y_1, \ldots, y_n) \rightarrow \neg B(x) \right)
\end{equation}

Here $y_1, \ldots, y_n$ are the variables free in $\exists x B(x)$. By the lemma, $T_1$ is a PA-theory. Obtain $T_2$ from $T_1$ by dropping the axiom $A_1 = \exists x B(x)$. From (4-1), we get by logic first $T_2 = B(f(y_1, \ldots, y_n))$ and then $T_2 = \not\exists x B(x)$. Therefore $T_1$ and $T_2$ are equivalent so that $T_2$ is a PA-theory. Next obtain $T_3$ from $T_2$ by replacing the axiom (4-1) by the two axioms

\begin{align}
(4-2) & \quad B(f(y_1, \ldots, y_n)) \\
(4-3) & \quad x < f(y_1, \ldots, y_n) \rightarrow \neg B(x)
\end{align}

Trivially $T_2$ and $T_3$ are equivalent. Finally obtain $T^*$ from $T_3$ by replacing (4-3) by one of its prenex forms. Then $T_2$ and $T^*$ are equivalent so that $T^*$ is a PA-theory. Let $(r, m')$ be the quantifier index of $T^*$. The difference between the set of axioms of $T$ and the set of axioms of $T^*$ is that $A_1$ has been replaced by (4-2) and the PNF of (4-3). The length of the quantifier strings of both these axioms is $l-1$. Therefore if $m = 1$, then $l = l-1$; and if $m > 1$, then $l = l$ and $m' = m-1$. In either case, $(r, m') < (l, m)$. This completes the proof of the induction step.

4.10 REMARK. In Shoenfield (1967), Statement (4-4) is claimed to be a provable result.

8-5 Quantum Logic

5.1 PRELIMINARIES. (I) If $H$ is a Hilbert space, then $L(H)$ denotes the set of subspaces of $H$.

(II) Let $L = (L(H), \emptyset, L, \perp, \wedge, \vee, =)$ be the lattice of subspaces of $H$ where

- $\emptyset$ is the null space
- $L = \{L(H)\}$
- $\perp = \text{the orthocomplement of } a$
- $\wedge = \text{the meet of } a \text{ and } b$
- $\vee = \text{the join of } a \text{ and } b$
- $a \leq b \iff a \vee b = b$

Then $L$ is an orthocomplemented lattice satisfying all the laws of quantum logic, including the orthomodular identity

\begin{equation}
(a \leq b \implies a = a \wedge b)
\end{equation}

5.2 DEFINITION. Let $a, b \in L(H)$.

(I) $a$ and $b$ are orthogonal, $a \perp b$, iff $a \leq b \perp$.
(II) a and b are compatible, \( a \leftrightarrow b \),

\[ \text{iff } (a \land b) \lor (b \land a) \lor a = a \lor b \]

5.3 **LEMMA.** Let \( a, b \in L(H) \). Then the following are equivalent:

1. \( a \leftrightarrow b \)
2. \( a \leftrightarrow b \)
3. There are pairwise orthogonal subspaces \( c, d, e \in L(H) \) such that
   \[ a = c \lor e, \quad b = d \lor e \]

5.4 **THEOREM.** Let \( a, b \in L(H) \) and \( P_a, P_b \) denote the projectors in \( H \) on \( a \) and \( b \), respectively. Then the following statements are equivalent:

1. \( a \leftrightarrow b \)
2. \( P_a \) and \( P_b \) commute.

**PROOF:**

1. \( \Rightarrow \) (2): Choose by Lemma 5.3 pairwise orthogonal \( c, d, e \in L(H) \) such that \( a = c \lor e \) and \( b = d \lor e \). Then
   \[ P_a P_b = P_{c \lor e} \quad P_{d \lor e} = (P_c + P_e)(P_d + P_e) = P_e \]
   \[ = (P_d + P_e)(P_c + P_e) = P_b P_a \]

2. \( \Rightarrow \) (1): Define
   \[ (5-2) \quad c = P_a P_b(H) = P_b P_a(H) = a \land b \]
   \[ (5-3) \quad c = a \land e \]
   \[ (5-4) \quad d = b \land e \]

Since \( c \leq a \) and \( c \leq b \), we get by the orthomodular identity (5-1),

\[ (5-5) \quad a = c \lor (c \land a) = c \lor e \]
\[ (5-6) \quad b = e \lor (e \land b) = e \lor d \]

Since \( c \leq e \) and \( d \leq c \), we have by Definition 5.2

\[ (5-7) \quad c \perp e \quad \text{and} \quad d \perp e \]

It remains to prove \( e \perp d \). From (5-3) and (5-4) follows

\[ (5-8) \quad P_c P_d(H) \leq c \perp d \]

By (5-2), (5-5), (5-6), and (5-7),

\[ P_c = P_a P_b = P_{c \lor e} P_{d \lor e} = (P_c + P_e)(P_d + P_e) = P_c + P_c P_d \]

Hence

\[ (5-9) \quad P_c P_d(H) \leq P_c(H) = e \]

which together with (5-8) implies

\[ P_c P_d(H) = \emptyset \]

Then

\[ (5-10) \quad c = P_c(H) \perp P_d(H) = d \]

Combining (5-5), (5-6), and (5-10), we see that \( a \leftrightarrow b \) by Lemma 5.3.

5.5 **REMARK.** Here the possible novelties occur in the proof of the direction (2) \( \Rightarrow \) (1).

5.6 **REMARK.** There are at least two possible ways of developing the foundations of quantum logic QL. One, initiated by Birkhoff and von Neumann, bases QL on the lattice structure of the class of subspaces of a Hilbert space. The other approach, initiated by among others Mackey and Jauch, study the general logic of expected outcomes of experiments. It turns out that this logic coincides with Birkhoff-von Neumann's QL.

In the next paragraphs, I expose the first steps in the development of the logic of experiments. The exposition is based on Cohen (1989). The purpose is to detect and, if possible, correct an error in Cohen's work.

5.7 **Experiments.** In physics, an experiment always consists in giving a well-defined input \( J \) to a well-defined system \( S \) and then observe which output \( O \) the input gives rise to. The output is called the outcome of the experiment. The basic principle in the set theoretic definition of experiments is to identify an experiment with its set of possible outcomes.

5.8 **DEFINITION.** (I) A **quasialgebra** \( Q \) is a nonempty collection of nonempty sets. The elements of \( Q \) are called **elements**. The elements of an experiment are the outcomes of the experiment.

(II) A set \( A \) is in \( Q \) if \( A \) is a subset of an experiment in \( Q \).

5.9 **DEFINITION.** Let \( Q \) be a quasialgebra.

(I) The two events \( A \) and \( B \) in \( Q \) are orthogonal, \( A \perp B \), if there is an experiment \( E \subset Q \) such that \( A, B \subset E \) and \( A \cap B = \emptyset \).

(II) A and \( B \) are orthogonal complements in \( Q \), \( A \perp B \), if \( A \cup B \subset Q \) is an experiment and \( A \perp B \).
5.10 DEFINITION. A manual \( \mathcal{M} \) is a quasi-manual which satisfies the axioms:

M1. If \( A, B, C, D \) are events in \( \mathcal{M} \) such that \( A \subset B, B \subset C, \) and \( C \subset D \), then \( A \subset D \).

M2. If \( E, F \in \mathcal{M} \) and \( E \subseteq F \), then \( E = F \).

M3. If \( x, y, z \) are outcomes in \( \mathcal{M} \) such that \( x \perp y, y \perp z, z \perp x \), then \( \{x, y, z\} \) is an event in \( \mathcal{M} \).

5.11 REMARK. The manual defined in §5.10 is intended to be a set theoretic representation which captures some essential features of a laboratory manual.

5.12 Reasons for the Axioms. I try to motivate the introduction of the axioms.

M1: Assume that the three experiments \( E = A \cup B, F = B \cup C, \) and \( G = C \cup D \) can be performed simultaneously. This is satisfied whenever \( B, C \subset \emptyset \). If \( A \) occurs in \( E \), then \( B \) does not and then \( C \) must occur in \( F \). Therefore \( D \) does not occur in \( G \). Similarly, if \( A \) does not occur in \( E \), then \( D \) occurs in \( G \). Therefore \( A \cap D = \emptyset \). We see that we can perform experiment \( G \) by doing \( E \), and vice versa. Thus there is in reality an experiment which tests the events \( A \) and \( D \) together. The axiom demands that this experiment should have a representation \( H = A \cup D \cup X \subset \mathcal{M} \). Then \( A \cap D \subseteq \emptyset \). Now we consider the case \( B = \emptyset \) or \( C = \emptyset \). Assume, e.g., \( B = \emptyset \). Then \( E = A \), \( F = C \), and \( F \subseteq G \). By Axiom M2, \( F = G \) so that \( D = \emptyset \). Then \( A \cup D \subseteq E \).

M2: This axiom says that experiments are maximal events. Let, e.g., \( F \) be an experiment where we can press button \( x \) or button \( y \) or do nothing \( \emptyset \). Then \( F = \{x, y, \emptyset\} \). Let \( E \) be the experiment where we can press button \( y \) or do nothing, i.e., \( E = \{y, \emptyset\} \). By Axiom M2, at most \( F \) can be in \( \mathcal{M} \). The reason for this may be that we cannot test for the outcomes \( y \) and \( \emptyset \) by performing \( F \). This seems not completely convincing to me. \( E \) and \( F \) in the example are clearly distinct experiments. In a few paragraphs, we shall see an unhappy consequence of Axiom M2.

M3: Let \( x, y \in E, y, z \in F, \) and \( x, z \in G \). Since the three experiments overlap, they can be performed simultaneously. Call the joined experi-
ment \( H \). Then \( \{x, y, z\} \) is an event in \( H \). Axiom M3 demands that this experiment \( H \) should be represented in \( \mathcal{M} \).

5.13 DEFINITION. Let \( \mathcal{E} \) be a collection of events in the manual \( \mathcal{M} \).
\( \mathcal{E} \) is compatible if \( \cup \mathcal{E} \) is an event in \( \mathcal{M} \). In particular, two events \( A \) and \( B \) in \( \mathcal{M} \) are compatible if \( A \cup B \) is an event in \( \mathcal{M} \).

5.14 DEFINITION. \( \mathcal{M} \) is a classical manual if every pair of events in \( \mathcal{M} \) is compatible.

5.15 OBSERVATION. \( \mathcal{M} \) is a classical manual iff \( \mathcal{M} \) contains exactly one experiment.

PROOF:
\( \Rightarrow \): Suppose that \( \mathcal{M} \) contains at least two different experiments \( E \) and \( F \). Since \( \mathcal{M} \) is classical, \( E \) and \( F \) are compatible events. Therefore \( E \cup F \) is an event in \( \mathcal{M} \). Let \( G \) be an experiment such that \( E \cup F \subset G \). Then \( E, F \subset G \). By Axiom M2, \( E = F = G \) contrary to the assumption.
\( \Leftarrow \): The inverse is trivial.

5.16 PROBLEM. The consequence of definitions 5.13-5.14 and Axiom M2 noted in the observation can hardly have been Cohen's intention. It seems easy to imagine laboratory manuals in classical physics which contain more than one experiment. I now consider two proposals which may improve the situation.

5.17 PROPOSAL I. The proof of the observation relies heavily on Axiom M2 in the definition of manuals. One possibility is to drop this axiom altogether from the definition. It is not known to the present writer whether it is still possible to develop the foundations of QL on the basis of experimental manuals without Axiom M2.

5.18 DEFINITION. (I) If \( \mathcal{M} \) is a manual, then \( \sigma[\mathcal{M}] = \cup \mathcal{M} \) is the class of all outcomes of experiments in \( \mathcal{M} \).

(II) Let \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) be manuals. An embedding of \( \mathcal{M}_1 \) in \( \mathcal{M}_2 \) is an injective mapping \( \Phi: \sigma[\mathcal{M}_1] \rightarrow \sigma[\mathcal{M}_2] \) such that for every experiment \( E \in \mathcal{M}_1 \), \( \Phi[E] = \{\Phi(x) \mid x \in E\} \) is an event in \( \mathcal{M}_2 \).
5.19 PROPOSAL 2. Another possibility is to keep Axiom M2 but instead drop Definition 5.14 and replace it by another definition of a classical manual. The idea behind the concept of a classical manual seems to be that it contains only experiments the outcomes of which can be tested in one single experiment, not necessarily that it actually is done. This idea could be expressed in the following definition.

DEFINITION 5.14 (alternative). Let $\mathcal{C}$ be a collection of manuals. $\mathcal{M}_1 \in \mathcal{C}$ is a classical manual in $\mathcal{C}$ if there is an $\mathcal{M}_2 \in \mathcal{C}$ such that $\mathcal{M}_1$ can be embedded in $\mathcal{M}_2$ and $\mathcal{M}_2$ contains exactly one experiment.

This definition implies a relativisation of classical manuals to the collection $\mathcal{C}$. A manual may be classical in one collection and nonclassical in another.

5.20 REMARK. Even Proposal 2 is not unproblematic. To see this, let $x$ and $y$ be two incompatible spin observables in a spin-1/2 particle $p$. Let

$$\mathcal{M}_1 = \{[x_{\text{up}}, x_{\text{down}}], [y_{\text{up}}, y_{\text{down}}]\}$$

We should want $\mathcal{M}_1$ to be a nonclassical manual in all contexts. It is, however, easy to construct a manual $\mathcal{M}_2$ as in Proposal 5.19:

$$\mathcal{M}_2 = \{[x_{\text{up}}, x_{\text{down}}, y_{\text{up}}, y_{\text{down}}]\}$$

It follows that $\mathcal{M}_1$ is classical in $\mathcal{C} = \{\mathcal{M}_1, \mathcal{M}_2\}$. It therefore seems that Proposal 5.19 is too liberal.

Though the manual $\mathcal{M}_2$ can be constructed in set theory, quantum theory does not allow such a manual. It would therefore seem that the notion of a classical manual cannot be defined within the measure set theoretic framework used by Cohen. An adequate distinction between classical and nonclassical manuals must be based on a mathematical model which more closely mimic the methods of real life experimental physics. In view of Remark 5.7, a better mathematical framework may be provided by automata theory and system theory. The present writer believes that this is also the way forward to a more adequate quantum logic than the one proposed by Birkhoff and von Neumann (1936). This point of view is developed in Hansen (1996).

8-6 Classical Geometry

6.1 REMARK. The present section contains a detailed, complete, and elementary proof of the circle of Apollonius. The proof is somewhat longer and somewhat different from the one given by Coxeter (1969). The present author believes that it nevertheless has some advantages.

6.2 PROBLEM. Let two fixed points $A$ and $A'$ in the Euclidean plane be given and $\mu \in \mathbb{R}_+$. Find the locus of all points $R$ such that

$A'R = \mu AR$

6.3 THEOREM (Apollonius). (I) If $\mu = 1$, the locus is the midnormal to $AA'$.

(II) If $\mu \neq 1$, the locus is the circle over $PP'$ as diameter where $P$ divides $AA'$ internally in the ratio $1:\mu$ and $P'$ divides $AA'$ externally in the ratio $1:\mu$. If $\mu > 1$, then $A$ is inside and $A'$ outside the circle. If $\mu < 1$, then $A$ is outside and $A'$ inside the circle.

PROOF: The case $\mu = 1$ is trivial. We now consider the case where $\mu > 1$. The theorem then follows for $\mu < 1$ by symmetry.

Let $P$ and $P'$ divide $AA'$ internally and externally in the ratio $1:\mu$. Thus

$$AP' = \mu AP, \quad AP'' = \mu AP'$$

Referring to Fig. 8-1, we must prove for an arbitrary point $R$ on the circle over $PP'$ with $Q$ as centre that

$$A'R = \mu AR, \quad A'Q = \mu AQ$$

where $Q$ is the second point of intersection between the prolongation of $A'R$ and the circle. We first compute the line segments on $AA'$ prolonged:

$$AA' = AP + PA', \quad AP + \mu AP = (1+\mu)AP$$

by (6-1), (6-1) and (6-2) together imply

$$\mu AP = A'P = AP + AA' = AP + (1+\mu)AP$$

Hence

$$AP'' = (\mu+1)AP(\mu-1)$$
(6-4) \[ PP' = AP + AP' = AP + (\mu+1) AP(\mu-1) = 2\mu AP(\mu-1) \]

(6-5) \[ OP = OP' = PP'/2 = \mu AP(\mu-1) \]

(6-6) \[ OA = OP - AP = \mu AP(\mu-1) - AP = AP(\mu-1) \]

The proof of

\[ A'R = \mu AR, \quad A'Q = \mu AQ \]

will be completed by the following lemmas 6.4-6.7.

**Proof:**

In Fig. 8-1, \( MM_2 \) is the midnormal to \( AA' \) and therefore it is the radical axis of the nonintersecting pencil of coaxial circles around \( A \) and \( A' \). The circle over \( PP' \) belongs to this pencil. By Lemma 6.4,

\[ M_2 A = M_2 T = M_2 A' \]

It follows that \( T, A, \) and \( A' \) all lie on the circle over \( A'T \) with \( M_2 \) as centre. Therefore \( A'TA \) is a straight-angled triangle. Thus

\[ \angle TAO = \angle A'TA = \angle A'TO \]

\[ AT \cdot A'T = OA \cdot OT = OA \cdot OP \]

By (6-5) and (6-6),

(6-7) \[ AT \cdot A'T = \mu \]

6.6 **Lemma.** \( \angle RAT = \angle QAT, \quad \angle PAR = \angle P'AQ \)

**Proof:**

Using similar triangles and Lemma 6.4, we see that

\[ M_1 A = M_1 A' = M_1 U \]

Then

\[ M_1 R \times M_1 Q = M_1 A^2 = M_1 U^2 \]

Since \( M, R', \) and \( Q' \) are projections on the prolongation of \( AA' \) of \( M_1, R, \) and \( Q, \) respectively, we can infer

\[ MR' \times MQ' = MA^2 \]

Therefore the circle over \( Q'R' \) as diameter also belongs to the coaxal pencil. It follows that \( R' \) and \( Q' \) divide \( AA' \) internally and externally in the same ratio:

\[ AR' : R'A' = AQ' : Q'A' \]

This may be rewritten as

(6-8) \[ AR' : AQ' = A'R' : A'Q' \]

Since \( A'Q'Q \sim A'R'R \),

\[ A'R' : A'Q' = RR' : QQ' \]

which together with (6-8) implies

\[ AR' : AQ' = RR' : QQ' \]
Therefore AR'R = AQ'Q and \( \angle PAR = \angle P'AQ \). It follows immediately that \( \angle RAT = \angle QAT \)

6.7 LEMMA. AR':A'R = AQ':A'Q = 1:1

PROOF:
Referring to Fig. 8-1, we have

\[
AR:AR' - AQ: AQ' = A'Q':A'Q = A'R:A'Q
\]

The first identity follows since AR'R = AQ'Q, the second since R' and Q' divide AA' internally and externally in the same ratio, and the third one follows since A'R'R = A'Q'Q. Hence

\[
(6-9) \quad AR:AR' = AQ:A'Q
\]

By Lemma 6.6, AR = AQ and AQ = AR'. Since

\[
AR \times AR' = AQ \times AQ = AT^2
\]

we get

\[
(6-10) \quad AR \times AQ = AT^2
\]

Similarly,

\[
(6-11) \quad A'R \times A'Q = A'T^2
\]

Divide (6-10) by (6-11) and use Lemma 6.5:

\[
(6-12) \quad (AR:AR') \times (AQ:A'Q) = (AT:A'T)^2 = 1:1
\]

By (6-9),

\[
AR:AR' = AQ:A'Q = 1:1
\]

References


9. Logical Foundations
of Logic Programming

9-1 Introduction

1.1 Systems for natural deduction, like PD in Hansen (1992, 1994), represent considerable progress towards a mechanisation of thinking. In principle, it is possible to perform deductions in a completely mechanical way in PD.

1.2 There are several motives for searching for a method of mechanical deduction in predicate logic:
(1) The ability to make logical inferences is an essential component of human intelligence. If we succeed in developing programs which can make deductions, we have made considerable progress toward artificial intelligence.
(2) The predicate logical languages which are studied in, e.g., chapters 6-12 of Hansen (1992, 1994) are presumably more "user close" than other formal languages in the sense that they are closer to the natural languages than other formal languages are. If we can use predicate logic as a programming language, the communication between man and machine should be eased considerably.

1.3 The difficulties in developing programs that make deductions in predicate logic turned out to be unexpectedly great. Not until the introduction of the programming language Prolog in the 1970's was an acceptable solution found. These problems were mainly the following.
(1) Predicate logical formulas are in general too complex to handle deductively for a computer. In automatic deductions, they lead to too large search spaces which the computer, in its search for a proof, cannot go through in a reasonable time. The solution is to limit oneself to sentences in clause form and especially, as in Prolog, to Horn clauses.
(2) Substitutions in connection with the deduction rules for V mostly give rise to too many variants which must be tested. The result is again a too big search space. The solution is found in the unification algorithm which gives an efficient and economical method for making substitutions.
(3) The fundamental deduction rules in PD and similar systems for the sentential logical operators also give rise to too big search spaces. They are in Prolog replaced by one single deduction rule, the resolution rule.

1.4 In the present survey article, we study the logical foundations of logic programming. Several logicians and computer scientists have contributed to the development of these foundations. Natural deduction was invented by Jankowski and Gentzen in the 1930's. Clauses were studied by Gentzen. Horn sentences were defined and investigated by Alfred Horn in the early fifties. The unification algorithm was formulated by Alan Robinson during the sixties on the basis of work by Herbrand. It was also Robinson who formulated the resolution method building on earlier work by Gentzen. Prolog was developed during the 1970's by Colmerauer, Kanoni, Panero, and Roussel, and by Kowalski among others.

As starting point, we take first-order predicate logic. With predicate logic as foundations, we develop systematically the logical foundations of logic programming, notably Prolog. In sections 2-5, clauses and Horn clauses are defined and methods for transforming from the standard form of predicate logic to clause form are developed. The three deduction rules needed in clause logic, namely resolution, unification, and contraction, are formulated and investigated. The road from clause logic to Prolog is indicated. Some weaknesses in the most common implementations of Prolog are pointed out. Section 6 contains a number of completeness results. In particular, general clause logic, Horn clause logic, and Prolog are all complete relative to the usual set theoretic semantics of predicate logic. All recursive functions can be calculated in Prolog so that Prolog as a programming language is at least as powerful as functional programming languages like those belonging to the ALGOL family. All problems which can be formulated and solved in predicate logic can, at least in principle, be formulated and solved in
general clause logic, Horn clause logic, and Prolog. Section 7 gives a model theoretic characterisation of Horn sentences and Horn clauses.

One purpose of the article is to show one possible road from logic to Prolog. The article should also make it possible for the programmer to understand Prolog and not only be able to use Prolog. The dominant tendency is otherwise that programmers use their programming languages without understanding them.

9-2 Clauses

2.1 DEFINITION. Let $P_1, \ldots, P_m, Q_1, \ldots, Q_n$ be atomic formulas in a predicate logical language. Then a clause is an expression of the form

$$P_1, \ldots, P_m \leftarrow Q_1, \ldots, Q_n \quad (m, n \geq 0)$$

(* In this chapter we use $P, Q, R, \ldots, P_1, P_2, \ldots$ to denote predicates and also as symbols in the metalanguage to denote atomic formulas.*)

2.2 REMARKS. (I) Let $x_1, \ldots, x_k$ be all the variables which occur in the atomic formulas $P_1, \ldots, P_m, Q_1, \ldots, Q_n$. Then the clause

$$P_1, \ldots, P_m \leftarrow Q_1, \ldots, Q_n$$

is just another way to write the sentence

$$\forall x_1 \ldots \forall x_k (Q_1 \land \ldots \land Q_n \rightarrow P_1 \lor \ldots \lor P_m)$$

This formula which is formed according to the formation rules for ordinary predicate logic is in standard form while the clause is said to be in clause form. We see that the universal quantifiers $\forall x_1 \ldots \forall x_k$ are tacitly understood in a clause. Though the quantifiers are not explicitly given, we always assume that all variables in a clause are bound by universal quantifiers.

(II) We see that the antecedent

$$Q_1, \ldots, Q_n$$

in the clause should be read as a conjunction $Q_1 \land \ldots \land Q_n$, while the consequent

$$P_1, \ldots, P_m$$

should be read as a disjunction $P_1 \lor \ldots \lor P_m$.

(III) In the literature on logic programming and Prolog, it is common to use the backward arrow '←' in clauses instead of the forward arrow '→' which is normally used in the standard form. We write, e.g.,

$$A \leftarrow B \quad ("A \text{ if } B")$$

and not

$$B \rightarrow A \quad ("B \text{ then } A")$$

The two expressions are nevertheless completely synonymous. The reason given by computer scientists for using '←' is that they want to stress the consequent by placing it first.

2.3 DEFINITION. Let $T'$ denote an arbitrary logically true formula. Let $\bot$ denote an arbitrary logically false formula.

(* '⊥' is read as 'false'. 'T' is read as 'true'.*)

2.4 EXAMPLE. (I) In a conjunction

$$Q_1 \land \ldots \land Q_n$$

we may have $n = 0$. We then have an empty conjunction. An empty conjunction is always true, i.e., equivalent with $T$. For we have the following truth condition for the conjunction

$$Q_1 \land \ldots \land Q_n \text{ is true} \iff \forall X (X \text{ is a component of the conjunction } \rightarrow X \text{ is true})$$

If the conjunction is empty, then the antecedent

$$X \text{ is a component of the conjunction}$$

is false for all formulas $X$. Then the implication, and as a consequence the truth condition, will be true. Therefore the empty conjunction is necessarily true.

(II) In a disjunction

$$P_1 \lor \ldots \lor P_m$$

we may have $m = 0$. Then the disjunction is empty. An empty disjunction is always false, i.e., equivalent with $\bot$. For we have the following falsity condition for the disjunction

$$P_1 \lor \ldots \lor P_m \text{ is false} \iff$$
\( \forall X \) (\( X \) is a component of the disjunction \( \rightarrow X \) is false)

In analogy with the reasoning above, we see that the empty disjunction always must be false.

2.5 Definition. Let \( P, Q_1, \ldots, Q_n \) be atomic formulas and let \( x_1, \ldots, x_k \) be the variables which occur in at least one of the atomic formulas.

(I) A Horn clause is a clause

\[ P \leftarrow Q_1, \ldots, Q_n \]

or

\[ \leftarrow Q_1, \ldots, Q_n \]

having at most one atomic formula in the consequent.

(II) A Horn sentence is a sentence of the form

\[ \forall x_1 \ldots \forall x_k (Q_1 \land \ldots \land Q_n \rightarrow P) \]

or

\[ \forall x_1 \ldots \forall x_k (Q_1 \land \ldots \land Q_n \rightarrow \bot) \]

(* A Horn sentence is the standard form of a Horn clause. *)

2.6 Example. We take as our starting point the general clause

\[ P_1, \ldots, P_m \leftarrow Q_1, \ldots, Q_n \]

with the standard form

\[ \forall x_1 \ldots \forall x_k (Q_1 \land \ldots \land Q_n \rightarrow P_1 \lor \ldots \lor P_m) \]

(I) Taking \( m = 1 \), we get the Horn clause

\[ P \leftarrow Q_1, \ldots, Q_n \]

A Horn clause is a clause having a categorical, non-disjunctive consequent

(II) If we take \( m = 0 \), we get a Horn clause with an empty disjunction in the consequent

\[ \leftarrow Q_1, \ldots, Q_n \]

According to Example 2.4, this may be read

\[ Q_1 \land \ldots \land Q_n \rightarrow \bot \]

which is equivalent with the standard form

\[ \neg (Q_1 \land \ldots \land Q_n) \]

and with (* use De Morgan *)

\[ \neg Q_1 \lor \ldots \lor \neg Q_n \]

(* In the sequel, we sometimes omit universal quantifiers from the standard form for the sake of simplicity. They are tacitly understood, exactly as in the case of the clause form. *)

If further \( n = 1 \), we get the clause

\[ \leftarrow Q \]

with standard form

\[ \neg Q \]

We see that clauses with empty consequent makes it possible to express negation in clause form.

(III) If we take \( n = 0 \), we get an empty conjunction in the antecedent

\[ P_1, \ldots, P_m \leftarrow \]

Example 2.4 gives the interpretation

\[ T \rightarrow P_1 \land \ldots \land P_m \]

which is equivalent with the standard form

\[ P_1 \land \ldots \land P_m \]

We see that clauses with empty antecedent make it possible to express nonhypothetical disjunctions in clause form.

If further \( m = 1 \), we obtain the clause

\[ P \leftarrow \]

with standard form

\[ P \]

Thus it is possible in clause form to express categorical statements in spite of the fact that clauses are implications.

(IV) If we finally let \( m = n = 0 \), we get the empty clause

\[ \leftarrow \]

where both antecedent and consequent are empty. According to § 2.4, the empty clause expresses the sentence

\[ T \rightarrow \bot \]
which is equivalent with
\[ \bot \]
This is very important. If we in an indirect deduction consisting of sentences in clause form are able to deduce the empty clause \( \bot \), then this is equivalent to a deduction in standard form of \( \bot \).

2.7 EXAMPLE. The following are all clauses. (a), (d), (e), and (g) are Horn clauses.

(a) \( P(x,y) \leftarrow P(x,y), \ y = z \)
(b) \( P(x,y), \ Q(a) \leftarrow R(y) \)
(c) \( 0 = 2, \ 0 = 3 \leftarrow 0 = 1, \ 1 = 2, \ 2 = 3 \)
(d) \( \leftarrow P(x,y), \ P(y,x) \)
(e) \( \leftarrow 0 = y \)
(f) \( P(f(x)), \ Q(x) \leftarrow \)
(g) \( R(x,f(x),y) \leftarrow \)

As an exercise, the reader may rewrite clauses (a)-(g) in standard form.

2.8 We are now going to show how every sentence in the standard form can be represented by a clause or a finite set of clauses. Let \( A \) be a given sentence in standard form. The transformation of \( A \) into clause form is a three-step operation:

(1) Rewrite \( A \) in prenex normal form

\[ Q_1 x_1 \ldots Q_m x_n B \]

where \( Q_1 \) is '\( \forall \)' or '\( \exists \)' and \( B \) is quantifier free.

(2) Eliminate all existential quantifiers from \( Q_1 x_1 \ldots Q_m x_n B \) by the Skolem transform. We write the result as

\[ \forall y_1 \ldots \forall y_k B^{*} \]

where \( B^{*} \) is quantifier free.

(3) Translate \( B^{*} \) into clause form.

Thus the pattern we follow is:

Standard form \( \rightarrow \) prenex form \( \rightarrow \) Skolem form \( \rightarrow \) clause form

2.9 The usual method for transforming a sentence into PNF as given, e.g., in Chapter 9 of Hansen (1992, 1994) can be used for the first step. We now study the Skolem transform of a sentence in PNF.

2.10 DEFINITION. We generalise the definition of predicates and function symbols such that we allow 0-place symbols.

(1) A 0-place function symbol is a constant.

(2) A 0-place predicate is a sentential parameter denoting an atomic formula.

2.11 REMARK. (I) The motivation for part (I) of the definition is the following. If we in an n-place function symbol \( f(x_1, \ldots, x_n) \) insert names of individuals for the variables, we get a symbol denoting a definite individual in the domain. Such a substitution does not change a 0-place function symbol since it contains no free variable places. The symbol must therefore all the time have denoted a certain individual, just like a constant.

(II) Part (II) of the definition can be justified in the following way. If we in an n-place predicate \( P(x_1, \ldots, x_n) \) substitute names of individuals for the variables, then we get an atomic formula with a definite truth-value in the model, i.e., a sentence. Such a substitution does not change a 0-place predicate which therefore all the time must have had its truth-value in the model, just like a sentential parameter.

2.12 DEFINITION. Let a sentence \( Q_1 x_1 \ldots Q_m x_n B \) in PNF be given.

\( Q_1 x_1 \ldots Q_m x_n B \) is in Skolem form \( \leftrightarrow \) all quantifiers in the quantifier prefix are universal quantifiers, i.e., the sentence has the form \( \forall x_1 \ldots \forall x_n B \).

2.13 The Skolem Transform. Assume that \( Q_1 x_1 \ldots Q_m x_n B \) is not in Skolem form. Then the sentence can be transformed into Skolem form by the Skolem transform.

Take the leftmost existential quantifier in the quantifier prefix

\[ \forall x_1 \ldots \forall x_k \exists x_{k+1} \ldots Q_m x_n B(x_1, \ldots, x_n) \]

Choose a new k-place function symbol \( f \) which does not occur in \( B \). Eliminate the quantifier \( \exists x_{k+1} \) and substitute \( x_{k+1} = f(x_1, \ldots, x_k) \) in \( B \):

\[ \forall x_1 \ldots \forall x_k Q_m x_n B(x_1, \ldots, x_k, f(x_1, \ldots, x_k), x_{k+2}, \ldots, x_n) \]
The variables \( x_1, \ldots, x_k \) are those whose quantifiers appear to the left of \( \exists x_{k+1} \). If \( k = 0 \) such that \( \exists x_{k+1} \) is placed leftmost in the sentence, then \( x_{k+1} \) should be replaced by a constant. If \( k = 0 \), \( x_{k+1} \) is replaced by a Skolem constant. If \( k > 0 \), \( x_{k+1} \) is replaced by a Skolem function.

Continue in the same way with that existential quantifier which is now leftmost until all existence quantifiers have been eliminated. Then the final result is a sentence in Skolem form.

If we have a set of sentences in PNF
\[
S = \{ A_1, \ldots, A_n \}
\]
which we want to be Skolem transformed, then we must have as many new function symbols as there are existential quantifiers altogether in \( A_1, \ldots, A_n \). The same function symbol cannot be used as a Skolem function for two different sentences in \( S \).

Note that the Skolem transform only has been defined for sentences, not for open formulas.

2.15 EXAMPLE. Find Skolem transforms for the following sentences.

(a) \( \exists x \forall y \forall z P(x,y,z) \)
(b) \( \exists x \exists y \forall z P(x,y,z) \)
(c) \( \forall x \exists y \forall z P(x,y,z) \)
(d) \( \forall x \forall y \exists z P(x,y,z) \)
(e) \( \forall x \exists y \exists z P(x,y,z) \)
(f) \( \forall x \forall y \exists z (Q(x,z) \rightarrow R(y,z)) \)
(g) \( \{ \forall x \exists y \forall z P(x,y,z), \forall x \exists y \exists z P(x,y,z) \} \)

SOLUTION:

(a) \( \forall y \forall z P(a,y,z) \)
(b) \( \forall z P(b,c,z) \)
(c) \( \forall x \forall z P(x,f(x),z) \)
(d) \( \forall x \forall y \forall z P(x,y,g(x,y)) \)
(e) \( \forall x \forall z P(x,m(x),z) \)
(f) \( \forall x \forall y (Q(x,h(x,y)) \rightarrow R(y,h(x,y))) \)
(g) \( \{ \forall x \forall y P(x,f(x),y), \forall x \forall y P(x,g(x),y) \} \)

2.16 REMARKS. (I) The introduction of Skolem functions according to the method in § 2.13 is the way existential quantifiers are expressed in clauses. The idea is that if, e.g., \( \forall x \exists y \ A(x,y) \) is true, then we may take one such representative for the \( y \)'s, \( y = f(x) \). In a domain of numbers, it is, e.g., true that \( \forall x \exists y x < y \), i.e., for every number \( x \) there is a number \( y \) which is larger than \( x \). E.g., we may let \( y = x + 1 = S(x) \). We then get as a possible Skolem transform \( \forall x x < S(x) \). We see that the choice of the representative for \( y \) depends on \( x \). \( y \) must therefore be represented by a function of \( x \).

From the examples and exercises in Chapter 10 of Hansen (1994) it is clear that existential quantifiers to a much higher degree than universal quantifiers add to the complications of deductions in standard form. Since existential quantifiers do not occur in clauses, deductions are considerably simplified in logic programming.

(II) It is important to understand that a sentence and its Skolem transform normally are not logically equivalent. As an example we consider the sentence \( \exists x P(x) \) with the Skolem transform \( P(c) \). Then we have
\[
\begin{align*}
P(c) & \models \exists x P(x) \\
\exists x P(x) & \models P(c)
\end{align*}
\]
Counterexample: \( M = (\mathbb{M}, P, c) = (\{a, b\}, \{a\}, b) \)

Let \( A \) be a sentence in PNF and \( A^* \) its Skolem transform. Then it is always true that
\( A^* \models A \)
In almost all cases
\( A \models A^* \)
An exact logical relation between a sentence and its Skolem transform is given in Lemma 2.18.

(III) We may justify the Skolem transform intuitively in the following way. As an example we look at the sentence \( \forall x \exists y R(x,y) \) with the Skolem transform \( \forall x R(x,f(x)) \).

Trivially, the following is valid
\[
\forall x R(x,f(x)) \models \forall x \exists y R(x,y)
\]
Therefore every model of \( \forall x R(x,f(x)) \) is a model of \( \forall x \exists y R(x,y) \).

Assume that there is a model \( M = (\mathbb{M}, R) \) such that
\( M \models \forall x \exists y R(x,y) \)
Then for every \( a \in M \) there is a \( b \in M \) such that \( R(a, b) \). For every \( a \), we choose one such \( b = b_a \) and define a function \( f: M \rightarrow M \) by \( f(a) = b_a \). We may then expand \( \mathcal{M} \) to the model

\[
\mathcal{M}' = (M, R, f)
\]

Trivially we have

\[
\mathcal{M}' \models \forall x R(x, f(x))
\]

We may conclude that

\[
\forall x \exists y R(x, y) \text{ has a model} \iff \text{the Skolem transform } \forall x R(x, f(x)) \text{ has a model.}
\]

This idea will be used in Lemma 2.18.

2.17 LEMMA. Let \( S \) be a set of sentences.
\( S \) is consistent \( \iff \) \( S \) has a model.

**PROOF:**

\( \Rightarrow \) If \( S = \{A_1, \ldots, A_n\} \) is consistent, then

\[
A_1, \ldots, A_n \not\models \bot
\]

Then there is a model \( \mathcal{M} \) such that

\[
\mathcal{M} \models A_1, \ldots, \mathcal{M} \models A_n, \mathcal{M} \not\models \bot.
\]

Therefore \( \mathcal{M} \) is a model of \( S \).

\( \Leftarrow \) If \( S \) has a model and is inconsistent, then some contradiction \( B \land \neg B \) is true in the model which is impossible.

2.18 LEMMA. Let \( S \) be a set of sentences in PNF and \( S^* \) its set of Skolem transforms. Then
\( S \) is consistent \( \iff \) \( S^* \) is consistent.

**PROOF:**

Let \( S = \{A_1, \ldots, A_n\} \) and \( S^* = \{A_1^*, \ldots, A_n^*\} \) where \( A_i^* \) is the Skolem transform of \( A_i \). We prove the lemma by induction on the total number of existential quantifiers in the sentences in \( S \).

If no sentence in \( S \) contains existential quantifiers, then \( A_i^* = A_i \) for every \( i \) and thus \( S^* = S \). It follows trivially that

\( S \) has a model \( \iff \) \( S^* \) has a model.

Assume that \( S \) contains \((k+1)\) existential quantifiers and that the lemma has been proved for all sets of sentences containing a total number of \( k \) existential quantifiers. Let \( A_i \) contain at least one existential quantifier. Then

\[
A_i = \forall x_1 \ldots \forall x_m \exists y B(x_1, \ldots, x_m, y)
\]

Let \( f(x_1, \ldots, x_m) \) be the Skolem function which in \( A_i^* \) replaces \( y \). Let

\[
A_i^* = \forall x_1 \ldots \forall x_m B(x_1, \ldots, x_m, f(x_1, \ldots, x_m))
\]

In the same way as in Remark 2.16(III), we see that

\[
(2-1) \quad S = \{A_1, \ldots, A_i, \ldots, A_n\} \text{ has a model} \iff S' = \{A_1, \ldots, A_i^*, \ldots, A_n\} \text{ has a model.}
\]

\( S' \) contains \( k \) existential quantifiers. By the induction hypothesis, it follows that

\[
(2-2) \quad S' \text{ has a model} \iff (S')^* = \{A_1^*, \ldots, A_i^*, \ldots, A_n^*\} \text{ has a model.}
\]

Since \((A_1')^*\) and \( A_i^* \) are logically equivalent,

\[
(2-3) \quad (S')^* \text{ has a model} \iff S^* \text{ has a model}
\]

From \( (2-1) \), \( (2-2) \), and \( (2-3) \) follows immediately

\[
(2-4) \quad S \text{ has a model} \iff S^* \text{ has a model}
\]

From Lemma 2.17 and \( (2-4) \), we get

\[
(2-5) \quad S \text{ is consistent} \iff S^* \text{ is consistent.}
\]

2.19 DEFINITION. Let \( S = \{A_1, \ldots, A_m\} \), \( T = \{B_1, \ldots, B_n\} \) be sets of sentences. Then \( S \) and \( T \) are logically equivalent iff

\[
A_1, \ldots, A_m \models B_j \quad \text{for } 1 \leq j \leq n
\]

\[
B_1, \ldots, B_n \models A_i \quad \text{for } 1 \leq i \leq m
\]

2.20 LEMMA. For every sentence in Skolem form \( \forall x_1, \ldots, \forall x_k A \) there is a finite set \( S \) of clauses such that \( S \) is logically equivalent with \( \forall x_1, \ldots, \forall x_k A \).

**PROOF:**

Since \( A \) is quantifier-free, \( A \) is constructed from atomic formulas solely by means of the sentential logical connectives. We may therefore treat \( A \) as a formula of sentential logic where different atomic formulas are treated as different sentential parameters.
Rewrite A in conjunctive normal form.

\[ A \iff C_1 \land \ldots \land C_n \]

Every \( C_i \) is a disjunction of negated and unnegated atomic formulas

\[ C_i = D_1 \lor \ldots \lor D_{10} \]

Reorder the disjuncts in \( C_i \) such that the negated atomic formulas prece the unnegated atomic formulas

\[ C_i \iff \neg P_1 \lor \ldots \lor \neg P_r \lor Q_1 \lor \ldots \lor Q_b \]

Applying the equivalences

\[ B \lor C \iff \neg B \rightarrow C \]

\[ \neg (\neg B_1 \lor \ldots \lor \neg B_b) \iff B_1 \land \ldots \land B_b \quad \text{(De Morgan)} \]

we have

\[ C_i \iff \neg (\neg P_1 \lor \ldots \lor \neg P_r) \rightarrow Q_1 \lor \ldots \lor Q_b \]

\[ \iff P_1 \land \ldots \land P_r \rightarrow Q_1 \lor \ldots \lor Q_b \]

\( C_i \) is therefore logically equivalent to the clause

\[ Q_1 \ldots Q_b \rightarrow P_1 \ldots P_r \]

We see that \( A \) is equivalent with a set consisting of \( n \) clauses, one clause for each of \( C_1, \ldots, C_n \).

2.21 THEOREM. Let \( S \) be a set of sentences in standard form. Then there is a set \( S_k \) of clauses such that \( S \) is consistent \( \iff \) \( S_k \) is consistent

PROOF:

Rewrite every sentence in \( S \) in prenex normal form. Call the obtained set of PNF sentences \( S_p \). Then \( S \) and \( S_p \) are logically equivalent. Hence

\[ S \text{ is consistent } \iff S_p \text{ is consistent} \]

Transform the sentences in \( S_p \) into Skolem form. Let \( S_b \) be the set of Skolem transforms of \( S_p \). By Lemma 2.18,

\[ S_p \text{ is consistent } \iff S_b \text{ is consistent} \]

According to Lemma 2.20 there is a set of clauses \( S_k \) such that \( S_b \) and \( S_k \) are logically equivalent. Then

\[ S_k \text{ is consistent } \iff S_k \text{ is consistent} \]

From (2-6), (2-7), and (2-8), the theorem follows immediately.

2.22 REMARKS. (I) Theorem 2.21 is the fundamental theorem of logic programming. It shows that clauses, in a certain sense, have the same expressive power as the standard form of predicate logic in spite of the fact that the class of clauses only is a fragment of predicate logic. Every sentence of predicate logic can be transformed into clause form.

The theorem also shows that every problem which can be formulated in predicate logic has a formulation in clause form. Let \( A_1, \ldots, A_n, B \) be sentences in standard form. Then the problem whether \( B \) follows logically from \( A_1, \ldots, A_n \) can be given a formulation in clause form. Let \( S = \{ A_1, \ldots, A_n, \neg B \} \). Let \( S_k \) be the corresponding set of clauses as in the proof of the theorem. Then

\[ A_1, \ldots, A_n \vdash B \iff A_1, \ldots, A_n, \neg B \vdash \bot \]

\[ \iff S \text{ is inconsistent} \]

\[ \iff S_k \text{ is inconsistent} \quad (* \text{by Theorem 2.21} *) \]

\[ \iff S_k \vdash \bot \quad (* \text{since } \vdash \text{ is equivalent with } \bot \text{ } *) \]

The first three equivalences are based on well-known results and methods of elementary logic. The fourth equivalence is based on methods and results on logic programming which will be developed later in this chapter.

(II) The equivalence

\[ A_1, \ldots, A_n \vdash B \iff S_k \vdash \bot \]

gives one reason why logic programming is based on the indirect proof procedure. The reason is that the Skolem transform does not preserve logical equivalence.

2.23 PROBLEM. Transcribe a given set of informal sentences into clause form.

METHOD:

The proof of Theorem 2.21 gives a method:

1. First formalise every sentence in standard form, \( A \).
2. Transform each formalised sentence \( A \) to prenex normal form, \( A_p \).
3. Construct a Skolem transform \( A_s \) of every PNF sentence \( A_p \).
(4) Use, as in the proof of Lemma 2.20, conjunctive normal form (CNF) to transform the Skolem sentence $A_k$ into clause form. (*Note, however, the comment below.*)

2.24 COMMENT. In practice, it is mostly too slow and cumbersome to transform $A_k$ to clause form by using CNF as suggested in point (4) of Method 2.23. A faster, though less systematic method is to transcribe $A_k$ directly into clauses by using logical equivalences of sentential logic. In the next paragraph, some such equivalences which are often useful are given.

2.25 Equivalences. We take as our starting point a sentence in Skolem form

$$\forall x_1 \ldots \forall x_k A$$

We want to transform it to clause form. Our goal is to have $A$ transformed into an equivalent formula $A^*$ in one of the following four forms.

**Form 1:** $A^*$ is a disjunction of atomic formulas

$$P_1 \lor \ldots \lor P_n \quad (n \geq 1)$$

Then $A$ has the clause form

$$P_1 \ldots P_n$$

**Form 2:** $A^*$ is the negation of a conjunction of atomic formulas

$$\neg(P_1 \land \ldots \land P_n) \quad (n \geq 1)$$

Then $A$ gets the clause form

$$P_1 \ldots P_n$$

**Form 3:** $A^*$ is an implication of the form

$$P_1 \land \ldots \land P_m \rightarrow Q_1 \lor \ldots \lor Q_n \quad (m, n \geq 1)$$

where each $P_i$ and $Q_j$ is atomic. The corresponding clause is

$$Q_1 \ldots Q_n \leftarrow P_1 \ldots P_m$$

**Form 4:** $A^*$ is a conjunction

$$B_1 \land \ldots \land B_n$$

where each $B_j$ has one of the forms 1-3. Then $A$ correspond to a clauses. Each $B_j$ gives rise to a clause according to the treatment of the forms 1-3 above.

During the transformation of $A$ to one of the forms 1-4, standard equivalences of sentential logic can be used. The following simple equivalences are particularly useful.

(K1) $A \rightarrow B \iff \neg A \lor B$

(K2) $A \leftrightarrow B \iff (A \rightarrow B) \land (B \rightarrow A)$

(K3) $\neg \neg A \iff A$ (Law of double negation, DN)

(K4) $\neg(A \land B) \iff \neg A \lor \neg B$ (De Morgan)

(K5) $\neg(A \lor B) \iff \neg A \land \neg B$ (De Morgan)

(K6) $\neg(A \rightarrow B) \iff A \land \neg B$

(K7) $A \land (B \lor C) \iff (A \land B) \lor (A \land C)$ (Distributive law)

(K8) $A \lor (B \land C) \iff (A \lor B) \land (A \lor C)$ (Distributive law)

(K9) $A \rightarrow \neg B \iff \neg(A \land B)$

(K10) $\neg A \rightarrow B \iff A \lor B$

(K11) $\neg A \rightarrow \neg B \iff B \rightarrow A$

When the sentence in Skolem form from which we start is of the form

$$\forall x_1 \ldots \forall x_k (B \rightarrow C)$$

it is often a good idea to keep the implication sign and let it be the main operator in one or more clauses. $B$ and $C$ must in this case be transformed in such a way that one of the forms 1-4 arises. In this process, the following equivalences may be useful.

(K12) $A \rightarrow B \land C \iff (A \rightarrow B) \land (A \rightarrow C)$

(K13) $A \lor B \rightarrow C \iff (A \rightarrow C) \land (B \rightarrow C)$

(K14) $A \land \neg B \rightarrow C \iff A \rightarrow B \lor C$

(K15) $A \rightarrow \neg B \lor C \iff A \land B \rightarrow C$

(K16) $A \rightarrow (B \rightarrow C) \iff A \land B \rightarrow C$ (Importation-exportation)

(K17) $(A \rightarrow B) \rightarrow C \iff (A \lor C) \rightarrow (B \rightarrow C)$
Clause form:

\[
\begin{align*}
\text{\textit{Example 2.31:}} &\quad \text{Not everybody owns a car.} \\
\text{SOLUTION:} &\quad \text{Standard form:} \\
&\quad \neg \forall x (P(x) \rightarrow \exists y (C(y) \land O(x,y))) \\
\text{Prenex form:} &\quad \exists x \forall y (P(x) \land (C(y) \rightarrow \neg O(x,y))) \\
\text{Skolem form:} &\quad \forall y (P(p) \land (C(y) \rightarrow \neg O(p,y))) \\
\text{Clause form:} &\quad \text{We apply (K21) and (K19) in order to simplify to} \\
&\quad P(p) \land \forall y (C(y) \rightarrow \neg O(p,y)) \\
&\quad \text{we now transform the second conjunct:} \\
&\quad C(y) \rightarrow \neg O(p,y) \iff \neg C(y) \lor \neg O(p,y) \quad (* \text{(K1)} *) \\
&\quad \iff \neg (C(y) \land O(p,y)) \quad (* \text{De Morgan} *) \\
\text{We get two clauses:} &\quad P(p) == \\
&\quad C(y), O(p,y) \\
\text{Example 2.32:} &\quad \text{Churchill will become prime minister if all members of the parliament vote for him.} \\
\text{SOLUTION:} &\quad \text{Standard form:} \\
&\quad \forall x (M(x) \rightarrow V(x,c)) \rightarrow P(x) \\
\text{Prenex form:} &\quad \exists x ((M(x) \rightarrow V(x,c)) \rightarrow P(x)) \\
\text{Clause form:} &\quad (M(m) \rightarrow V(m,c)) \rightarrow P(c) \\
\text{(K17) yields the equivalent sentence} \\
&\quad (M(m) \lor P(c)) \land (V(m,c) \rightarrow P(c)) \\
\text{We obtain the clauses} &\quad M(m), P(c) \leftrightarrow \\
&\quad P(c) \leftrightarrow V(m,c) \\
\text{Sentences having an implicative antecedent often occur and are} \\
\text{natural in informal language and in the standard form. On the other} \\
\text{hand, it is difficult to see that the two clauses express a sentence with} \\
\text{implicative antecedent. This is one of the points where the clause form} \\
\text{is much less natural and intuitive than the standard form.} \\
\text{Example 2.33 (Implicative antecedent).} &\quad \text{An exam where all answers are correct will pass.} \\
\text{SOLUTION:} &\quad \text{Standard form:} \\
&\quad \forall x (E(x) \land \forall y (A(y,x) \rightarrow C(y)) \rightarrow P(x)) \\
\text{Prenex form:} &\quad \forall x \exists y (E(x) \land (A(y,x) \rightarrow C(y)) \rightarrow P(x)) \\
\text{Skolem form:} &\quad \forall x (E(x) \land (A(a(x),x) \rightarrow C(a(x)))) \rightarrow P(x)) \\
\text{Clause form:} &\quad \text{We apply (K18)} \\
&\quad (E(x) \rightarrow A(a(x),x) \lor P(x)) \land (E(x) \land C(a(x)) \rightarrow P(x)) \\
\text{This gives the following two clauses} &\quad A(a(x),x), P(x) \leftrightarrow E(x) \\
&\quad P(x) \leftrightarrow E(x), C(a(x))
2.34 EXAMPLE (Conjunctive disjuncts). Either Stemson is promoted and gets a raise in salary or else he will quit and apply for another job.

SOLUTION:
Standard form:
\[(P(x) \land R(x)) \lor (Q(x) \land A(x))\]

Clause form:
We rewrite the sentence so that it becomes a conjunction of disjunctions. We use the distributive laws (K7) and (K8).
\[(P(x) \lor Q(x)) \land (R(x) \lor Q(x)) \land (P(x) \lor A(x)) \land (R(x) \lor A(x))\]

This gives rise to the following four clauses:
- \(P(x), Q(x) \rightarrow\)
- \(R(x), Q(x) \rightarrow\)
- \(P(x), A(x) \rightarrow\)
- \(R(x), A(x) \rightarrow\)

This is another point where the clause form is less natural than the standard form.

2.35 EXAMPLE (Definitions). We use the following symbols:
- \(G(x,y)\): x is paternal grandmother of y
- \(M(x,y)\): x is mother of y
- \(F(x,y)\): x is father of y

Then the relation 'x is paternal grandmother of y' may be defined by
\[\forall x \forall y (G(x,y) \leftrightarrow \exists z (M(x,z) \land F(z,y)))\]

Write this definition in clause form.

SOLUTION:
By means of (K2) and (K22), the definition is divided into two parts.
The if-part:
\[\forall x \forall y (\exists z (M(x,z) \land F(z,y))) \rightarrow G(x,y)\]
The only if-part:
\[\forall x \forall y (G(x,y) \rightarrow \exists z (M(x,z) \land F(z,y)))\]

(1) First we treat the if-part.

Prenex form:
\[\forall x \forall y \forall z (M(x,z) \land F(z,y)) \rightarrow G(x,y)\]

Clause form:
\[G(x,y) \leftrightarrow M(x,z), F(z,y)\]

(2) Next we examine the only if-part:
Prenex form:
\[\forall x \forall y \exists z (G(x,y) \rightarrow M(x,z) \land F(z,y))\]

Skolem form:
\[\forall x \forall y (G(x,y) \rightarrow M(x,f(x,y)) \land F(f(x,y),y))\]

Clause form:
- \(M(x,f(x,y)) \leftrightarrow G(x,y)\)
- \(F(f(x,y),y) \leftrightarrow G(x,y)\)

(3) The definition is thus represented by three Horn clauses:
\[G(x,y) \leftrightarrow M(x,z), F(x,y)\]
\[M(x,f(x,y)) \leftrightarrow G(x,y)\]
\[F(f(x,y),y) \leftrightarrow G(x,y)\]

The if-part is used to prove that the paternal grandmother-relation obtains between two individuals. The only if-part is used to prove that the paternal grandmother-relation does not obtain between two individuals. In logic programs, only the if-part is included while the only if-part is omitted. The answer that the paternal grandmother-relation does not obtain between two individuals, the program arrives at by failing to prove that the paternal grandmother-relation is true of the two individuals.

2.36 EXAMPLE. (i) We use the symbol
- \(P(x)\): x is a prime number

In the domain of positive integers, prime numbers are defined:
\[\forall x (P(x) \leftrightarrow \forall y \forall z (y \neq x \land y = z \lor z = x))\]

("According to the definition, 1 is counted as a prime.")

Write the definition in clause form.
(II) Goldbach's conjecture in number theory states that every positive even number can be written as a sum of two prime numbers, i.e.,
\[ \forall x \ (\exists y \ y = 2x \rightarrow \exists z \exists v \ (P(x) \land P(y) \land x = y + v)) \]
where the predicate \( P(x) \) is defined as in (a).
Express Goldbach's conjecture in clause form.

2.37 Horn clauses. Horn clauses constitute a proper subset of the class of clauses. It is not all sentences in standard form which can be written as Horn clauses. We have seen that a consequence of Theorem 2.22 is that every problem which can be formulated and solved in predicate logic also can be formulated and solved by clauses. In view of the smaller expressive power of Horn clauses, it seems reasonable to conjecture that the same does not hold good of them. Somewhat surprisingly it turns out that this conjecture is false. All problems which can be formulated and solved in predicate logic can also be formulated and solved in Horn clauses. A proof of this result occurs in Section 6 of the present chapter. The result is a consequence of the fact that all computable functions can be expressed and computed in Horn clauses. Therefore all algorithms can be expressed and executed in Horn clauses. Unfortunately, these methods lead to logic programs which it takes extremely long time to execute compared with logic programs which are formulated by direct methods.

In theory, though not in practice, Horn clauses suffice for all problems. There are, however, a simple device by which clauses that are not in Horn form in many cases can be replaced by Horn clauses, namely by replacing negated predicates by new negative predicates.

2.38 EXAMPLE (Negative predicates). The sentence
Every number is either even or odd
has the clause form
\[ E(x), O(x) \leftarrow N(x) \]
which is not in Horn form. Its standard form is
\[ N(x) \rightarrow E(x) \lor O(x) \]
The equivalence (K14) yields
\[ N(x) \land \neg E(x) \rightarrow O(x) \]
We introduce a new negative predicate
\[ U(x) : x \text{ is non-even} \]
instead of the negated predicate \( \neg J(x) \) and get the Horn clause
\[ O(x) \leftarrow N(x), U(x) \]
i.e., every number which is non-even is odd.

2.39 REMARK. The two clauses
(1) \[ B(x), O(x) \leftarrow N(x) \]
(2) \[ O(x) \leftarrow N(x), U(x) \]
in Example 2.38 are not logically equivalent. To prove (1) and (2) equivalent, we need two further assumptions, namely the assumption
\[ \neg (U(x) \land E(x)) \]
i.e., nothing is both non-even and even at the same time, which has the clause form
(3) \[ \leftarrow U(x), E(x) \]
and the assumption
\[ U(x) \lor E(x) \]
i.e., a number is either non-even or even, which has the clause form
(4) \[ U(x), E(x) \leftarrow \]
The clause (3) is in Horn form and can be used as a program sentence in a program containing only Horn clauses. On the other hand, clause (4) is not in Horn form. If we limit ourselves to Horn clauses and use the method of negative predicates in Example 2.38, we lose the information in clause (4). For many problems this piece of information is unimportant. In such cases the method of negative predicates may be used to advantage.

2.40 REMARK. To wind up, we repeat the pattern we follow when we write a given informal sentence in clause form. First we formalize the sentence in standard form and then follow the route
standard form \( \Rightarrow \) prefix form \( \Rightarrow \) Skolem form \( \Rightarrow \) clause form.
9-3 Resolution

3.1 Deduction. It is possible to make deductions where the conclusion and all the premises are sentences in clause form. It turns out that in this connection only three deduction operations and rules are needed:

- (1) Resolution.
- (2) Unification.
- (3) Contraction.

For deduction in Horn clauses it suffices with resolution and unification. Resolution is a principle of sentential logic. It will be treated in this section. Unification is an operation of predicate logic. Contraction is a combination of unification and an operation of sentential logic. They will be treated in Section 4.

Deduction in clause form is mechanically simpler than deduction in general predicate logic. For a computer it is therefore almost always much easier to find a correct deduction when we limit ourselves to the realm of clause forms. This is in particular true for the still more limited class of Horn clauses. In contrast, deductions in clauses are less suited for the human intellect. We do not have the same help of our logical and linguistic intuition as in connection with the standard form.

3.2 The Resolution Rule. Let \( P, Q, R, S, T \) be atomic formulas. Then the resolution rule may be formulated:

- (1) \( P_1 \wedge \cdots \wedge P_m \wedge T \leftarrow Q_1 \wedge \cdots \wedge Q_n \)
- (2) \( R_1 \wedge \cdots \wedge R_p \leftarrow S_1 \wedge \cdots \wedge S_q \wedge T \)
- (3) \( P_1 \wedge \cdots \wedge P_m \wedge R_1 \wedge \cdots \wedge R_p \leftarrow Q_1 \wedge \cdots \wedge Q_n \wedge S_1 \wedge \cdots \wedge S_q \)

(Resolution)

(\(^*\) The two occurrences of \( T \) are eliminated. The consequents in (1) and (2) are put together. The antecedents in (1) and (2) are put together.\(^*\) )

\( T \) may be placed anywhere in the consequent of (1) and anywhere in the antecedent of (2). The purpose with the placement \( T \) has been given in the above formulation is only to get a simple and easy-to-grasp form of the resolution rule.

(1) and (2) are the resolution premises. (3) is called the resolvent. Applying the resolution rule is called to resolve.

3.3 EXAMPLE. Rewritten in standard form, the resolution rule gets the following form:

\[
\begin{align*}
(1) & \quad Q_1 \wedge \cdots \wedge Q_n \rightarrow P_1 \lor \cdots \lor P_m \lor T \\
(2) & \quad S_1 \wedge \cdots \wedge S_q \wedge T \rightarrow R_1 \lor \cdots \lor R_p \\
(3) & \quad Q_1 \wedge \cdots \wedge Q_n \wedge S_1 \wedge \cdots \wedge S_q \rightarrow P_1 \lor \cdots \lor P_m \lor R_1 \lor \cdots \lor R_p
\end{align*}
\]

Using methods of sentential logic, e.g., the short cut method, it is easy to show that the conclusion (3) is a sentence logical consequence of the premises (1) and (2). (\(^*\) This result implies that the resolution rule is logically sound.\(^*\) )

3.4 Horn Clauses and Resolution. If we limit ourselves to Horn clauses, the resolution rule gets a simpler form. We must then have \( m = 0 \) and either \( p = 0 \) or \( p = 1 \).

For Horn clauses, we get three main forms of resolution which will be called:

- middle-out resolution,
- forward resolution,
- backward resolution.

3.5 Middle-out Resolution. In the resolution rule of § 3.2, we set \( m = 0 \), \( n > 0 \) and \( p = 1 \). The resolution rule then assumes the following form:

- (1) \( T \leftarrow Q_1, \ldots, Q_n \)
- (2) \( R \leftarrow S_1, \ldots, S_q, T \)
- (3) \( R \leftarrow Q_1, \ldots, Q_n, S_1, \ldots, S_q \)

3.6 Forward Resolution. In the resolution rule in § 3.2, we set \( m = 0 \), \( n = 0 \) and \( p = 1 \). The resolution rule now assumes the following form:

- (1) \( T \leftarrow \)
- (2) \( R \leftarrow S_1, \ldots, S_q, T \)
- (3) \( R \leftarrow S_1, \ldots, S_q \)

(1) expresses the categorical statement \( T \). (\(^*\) Forward resolution is also called bottom-up resolution in English.\(^*\) )

3.7 Backward Resolution. In the resolution rule in § 3.2, we let \( m = 0 \) and \( p = 0 \):
(1) \( T \leftarrow Q_1, \ldots, Q_n \)

(2) \( \leftarrow S_1, \ldots, S_m, T \)

(3) \( \leftarrow Q_1, \ldots, Q_m, S_1, \ldots, S_n \)

Backward resolution can be applied when one of the premises (in the present case (2)) is a negation, i.e., has an empty consequent. In Prolog this is the only form of resolution applied. (* Backward resolution is in English also called top down resolution.*)

3.8 EXAMPLE 3 I Modus ponens (or \((\rightarrow\text{E})\)) is the inference

\[ P \rightarrow Q, \ P \vdash Q \text{ (MPP)} \]

In clause form, it has the form

\[ Q \leftarrow P \]

\[ P \leftarrow \]

\[ Q \leftarrow \]

We get the following deduction of (MPP) by resolution:

(1) \( Q \leftarrow P \) \hspace{1cm} \text{Premiss}

(2) \( P \leftarrow \) \hspace{1cm} \text{Premiss}

(3) \( Q \leftarrow \) \hspace{1cm} \text{Premiss,Resolution}

We have applied forward resolution. We see that \((\rightarrow\text{E})\) is implicit in the resolution rule as a special case.

(II) Modus tollens (MTT) is the inference

\[ P \rightarrow Q, \neg Q \vdash \neg P \text{ (MTT)} \]

In clause form, (MTT) has the form

\[ Q \leftarrow P \]

\[ \neg Q \leftarrow \]

\[ P \leftarrow \]

We give a resolution proof of (MTT):

(1) \( Q \leftarrow P \) \hspace{1cm} \text{Premiss}

(2) \( \neg Q \leftarrow \) \hspace{1cm} \text{Premiss}

(3) \( P \leftarrow \) \hspace{1cm} \text{Premiss,Resolution}

We have used backward resolution. We see that (MTT) is a special case of backward resolution.

(III) The Syllogism Principle is the inference

\[ P \rightarrow Q, \ Q \rightarrow R, \ 1- \ P \rightarrow R \text{ (Syllogism Principle)} \]

Its clause form is

\[ Q \leftarrow P \]

\[ R \leftarrow Q \]

\[ R \leftarrow P \]

A possible resolution proof is

(1) \( Q \leftarrow P \) \hspace{1cm} \text{Premiss}

(2) \( R \leftarrow Q \) \hspace{1cm} \text{Premiss}

(3) \( R \leftarrow P \) \hspace{1cm} \text{1,2,Resolution}

We see that the syllogism principle is a special case of middle out resolution.

(IV) All the basic sentence logical deduction rules of the system SD in Hansen (1994) can be shown to be implicit in the resolution rule if we assume that A, B and C in the deduction rules of SD represent atomic formulas. Almost all sentence logical deduction problems in clause form can be solved by resolution alone. In rare cases also the contraction rule described in Section 4 is needed. It can be proved that deduction based on resolution and contraction is complete for sentential logic in clause form.

3.9 Indirect Deduction. In logic programming, we always and only use indirect deduction. It implies that one writes the negation of the desired conclusion in clause form, adds these clauses as extra premises, and tries to derive \( \bot \). There are two reasons for this choice.

(1) Indirect derivation always works, both in standard form and in clause form, given that there exists a deduction for the given problem. In contrast, there are solvable deduction problems where direct deduction does not work.

(2) The other reason has to do with the fundamental theorem 2.21 of logic programming. If \( S \) is a set of sentences in standard form and \( S^* \) is the result of writing the sentences in \( S \) in clause form, then \( S \) and \( S^* \) are normally not logically equivalent. But Theorem 2.21 shows that \( S \) is inconsistent iff \( S^* \) is inconsistent, i.e.,
3.11 EXAMPLE (Backward resolution). Show by the clause method
\[ P \rightarrow Q, \ Q \rightarrow R, \ P \vdash R \]

**SOLUTION:**

(I) We write the premises in clause form.
\[ Q \leftarrow P; \ R \leftarrow Q; \ P \leftarrow \]

We write \(-R\) in clause form.
\[ \leftarrow R \]

(* Note that all the clauses are of Horn type.*)

(II) We are now in a position to deduce \(-\)

\[
\begin{array}{ll}
(1) & Q \leftarrow P \\
(2) & R \leftarrow Q \\
(3) & P \leftarrow \\
(4) & \leftarrow R \\
(5) & \leftarrow Q \\
(6) & \leftarrow P \\
(7) & \leftarrow \\
\end{array}
\]

3.12 REMARK. (I) In all the three applications in lines \((6)-(8)\), we use backward resolution according to § 3.7. This is characteristic of deductions where all the premises are Horn clauses. The only form of resolution needed is backward resolution. Such a deduction will be called a *backward deduction*.

(II) We start from behind with a clause of the form
\[ \leftarrow P_1, \ldots, P_n \]

Step by step we move backward until we arrive at \(-\). In the example, we have the three premises \((1)-(3)\). In Prolog, they are called the *program sentences*. We have the negated conclusion \((4)\). In Prolog, it is called the question or the goal. We ask whether R is valid by feeding in \(-R\), i.e., \(-R\). If the system succeeds in deducing \(-\), the answer is "Yes, R is valid!" If the system fails to deduce \(-\), the answer is "No, R is not valid!"

(III) We start from behind with the negation of the conclusion, \(-R\). Then we search among the three program sentences to find a clause having consequent R.
(a) If there is no such clause, the deduction halts and \( \leftarrow \) is not deducible.

(b) Is there exactly one clause with \( R \) as a consequent as in the example, namely \( R \leftarrow Q, \leftarrow R \) is resolved with \( R \leftarrow Q \) and we get \( \leftarrow Q \). Then another search is made among the premises for a clause with the consequent \( Q \). Etc.

(c) If there are more than one clause among the program sentences having the consequent \( R \), one must use trial and error and resolve each of them with \( \leftarrow R \) to see if one of the alternatives leads to a deduction of \( \leftarrow \).

3.13 Deductive Structure. A backward deduction in Horn clauses has the following standard form.

(1) First occurs a number of premises (program sentences) in one of the forms

\[
Q \leftarrow P_1, \ldots, P_n
\]

\[
\leftarrow P_1, \ldots, P_n
\]

(2) Next comes the negation of the desired conclusion in the form of one or more Horn clauses. They also belong to the set of premises.

(3) A premise of the form

\[
\leftarrow P_1, \ldots, P_n
\]

is resolved with another premise in which one of the \( P_i \) occur as the consequence. The result is a clause of the form

\[
\leftarrow S_1, \ldots, S_p
\]

(4) One continues this by every time backward resolving the clause obtained in the preceding line with some premis.

(5) Finally, the empty clause \( \leftarrow \) turns up and the deduction halts.

3.14 REMARK. (I) Prolog is logic programming in Horn clauses. Prolog uses consistently and exclusively backward resolution. In the sequel, we do the same for all deduction problems which are defined solely in terms of Horn clauses.

(II) In Prolog, all program sentences have the form

\[
Q \leftarrow P_1, \ldots, P_n
\]

The question always has the form

\[
\leftarrow R_1, \ldots, R_m
\]

Prolog therefore always works backward from the question to \( \leftarrow \).

(III) Deduction problems, in which non-Horn clauses occur among the premises, normally do not have a solution in the simple standard form of § 3.13. Such deductions demand at least one application of the general resolution rule of § 3.2.

(IV) The deduction problem in Example 3.11 can also be solved by forward deduction. Try that as an exercise!

3.15 EXAMPLE. Show

\[
P \land Q \rightarrow R \vdash (P \rightarrow R) \lor (Q \rightarrow R)
\]

SOLUTION:

(I) Clause form:

We write the premises in clause form:

\[
R \leftarrow P, Q
\]

According to Method 3.10, the conclusion must be negated:

\[
\neg((P \rightarrow R) \lor (Q \rightarrow R)) \iff \neg(P \rightarrow R) \land \neg(Q \rightarrow R)
\]

(* De Morgan *)

\[
\iff (P \land \neg R) \land (Q \land \neg R)
\]

(* K6 *)

\[
\iff P \land Q \land \neg R
\]

This gives rise to the following clauses:

\[
P \leftarrow
\]

\[
Q \leftarrow
\]

\[
\leftarrow R
\]

(II) Deduction:

(1)

\[
R \leftarrow P, Q
\]

P

(2)

\[
P \leftarrow
\]

P

(3)

\[
Q \leftarrow
\]

P

(4)

\[
\leftarrow R
\]

P

(5)

\[
\leftarrow P, Q
\]

1, 4, Resolution

(6)

\[
\leftarrow Q
\]

2, 5, Resolution

(7)

\[
\leftarrow
\]

3, 6, Resolution

The premises (1)-(4) are all in Horn form. We have only used backward resolution.
3.16 EXAMPLE. Show

\[ P \rightarrow ((Q \land R) \lor T), \quad Q \land R \rightarrow \neg P, \quad S \rightarrow \neg T \vdash P \rightarrow \neg S \]

SOLUTION:

(1) Clause form:

\[ P \rightarrow ((Q \land R) \lor T) \iff P \rightarrow (Q \lor T) \land (R \lor T) \]

\[ \iff (P \rightarrow Q \lor T) \land (P \rightarrow R \lor T) \quad (*) \text{(K12)} (*) \]

This gives rise to the clauses

(1) \( Q, T \leftarrow P \)

(2) \( R, T \leftarrow P \)

(3) \( Q \land R \rightarrow \neg P \iff \neg (Q \land R \land P) \quad (*) \text{(K9)} (*) \)

yields the clause

(3) \( \leftarrow Q, R, P \)

(4) \( S \rightarrow \neg T \iff \neg (S \land T) \quad (*) \text{(K9)} (*) \)

gives the clause

(4) \( \leftarrow S, T \)

We now take the negation of the conclusion:

\[ \neg (P \rightarrow \neg S) \iff P \land \neg S \quad (*) \text{(K6)} (*) \]

\[ \iff P \land S \quad (*) \text{(K3)} (*) \]

As a result, we have the following two clauses

(5) \( P \leftarrow \)

(6) \( S \leftarrow \)

II Deduction:

(1) \( Q, T \leftarrow P \)

(2) \( R, T \leftarrow P \)

(3) \( \leftarrow Q, R, P \)

(4) \( \leftarrow S, T \)

(5) \( P \leftarrow \)

(6) \( S \leftarrow \)

(7) \( \leftarrow T \)

(8) \( Q \leftarrow P \)

9-4 Unification

4.1 EXAMPLE. The following logical consequence relation is trivially correct:

\[ \forall x (P(x) \rightarrow Q(x)), \quad P(a) \vdash Q(a) \]

PROOF:

(1) \( \forall x (P(x) \rightarrow Q(x)) \)

(2) \( P(a) \)

(3) \( P(a) \rightarrow Q(a) \)

(4) \( Q(a) \)

As a result, we have the following two clauses

(5) \( P \leftarrow \)

(6) \( Q \leftarrow \)

However, it cannot be proved solely by the resolution rule.

(1) \( Q(x) \leftarrow P(x) \)

(2) \( P(x) \leftarrow \)

(3) \( \leftarrow Q(x) \)

\( \leftarrow Q(a) \) and \( \leftarrow Q(x) \leftarrow P(x) \) cannot be resolved with each other since the terms 'a' and 'x' in 'Q(a)' and 'Q(x)' are different. For the same reason, \( P(x) \leftarrow \) and \( \leftarrow Q(x) \leftarrow P(x) \) cannot be resolved. \( P(a) \leftarrow \) and \( \leftarrow Q(a) \) cannot be resolved with each other because the predicates 'P' and 'Q' are different. Clearly, we need to be able to make a substitution of 'a' for 'x' in 'Q(x) \leftarrow P(x)' corresponding to the one we made in line (3) in the
deduction in standard form when we applied (VE). We then get the following resolution proof.

1. \( Q(x) \leftarrow P(x) \)
2. \( P(x) \leftarrow \)
3. \( \leftarrow Q(a) \)
4. \( Q(a) \leftarrow P(a) \)
5. \( \leftarrow P(a) \)
6. \( \leftarrow \)

4.2 The \((\forall I)\)-Rule. A clause has the form

\[ P_1, ..., P_m \leftarrow Q_1, ..., Q_n \]

with the standard form

\[ \forall x_1 \cdots \forall x_k (Q_1 \land \cdots \land Q_n \rightarrow P_1 \lor \cdots \lor P_m) \]

As an example,

\[ P(f(y),z) \leftarrow Q(x) \]

is a clause with the standard form

\[ \forall x \forall y \forall z (Q(x) \rightarrow P(f(y),z)) \]

The only strictly predicate logical deduction rules needed in deductions in clause form are therefore counterparts of (VE) and (VI) since \( \exists \) does not occur in clauses. (The deduction rules for \( \forall \) will be treated later in this section.)

Since for every free variable in a clause it is understood that it is universally quantified, \((\forall I)\) is automatically applied in every new line in a deduction where a clause arises containing variables. Note that since every variable in a premise is universally quantified, the restriction for \((\forall I)\) is automatically satisfied. The \((\forall I)\)-rule is thus implicit in the clause form and performed automatically.

4.3 The \((\forall E)\)-Rule. The \((\forall E)\)-rule has the form

\[ \forall x A(x) \]

\[ \frac{A(t)}{A(t)} \]

if \( t \) is free for \( x \) in \( A(x) \).

Because universal quantifiers are not written in clauses, \((\forall E)\) here takes the form of a substitution of a term for a variable:

\( A(x) \)

\( A(t) \)

We first note that we in deductions in clause form need not care about the restriction on \((\forall I)\). Assume, e.g., that we have a clause

\[ P(x,y) \leftarrow Q(x,y) \]

with the standard form

\[ \forall x \forall y (Q(x,y) \rightarrow P(x,y)) \]

We want to make the substitution

\[ \{ x = f(x,y), y = g(x,y) \} \]

in the clause and get

\[ P(f(x,y),g(x,y)) \leftarrow Q(f(x,y),g(x,y)) \]

We cannot directly by \((\forall E)\) get the corresponding standard form formula from (2) since the term \( f(x,y) \) is not free for \( x \) in \( \forall y (Q(x,y) \rightarrow P(x,y)) \). But if we first change bound variables in (2)

\[ \forall x \forall y \forall z (Q(z,w) \rightarrow P(z,w)) \]

we may without any risk of conflict with the restriction use \((\forall E)\) twice to obtain

\[ Q(f(x,y),g(x,y)) \rightarrow P(f(x,y),g(x,y)) \]

which is the standard form of (3).

Applying \((\forall E)\) in clause form thus consists in making substitutions \( \{ x_1 = t_1, ..., x_n = t_n \} \):

\[ A(x_1, ..., x_n) \]

\[ \frac{A(t_1, ..., t_n)}{A(t_1, ..., t_n)} \]

(‘Note that here all occurrences of \( x_i \) in \( A(x_1, ..., x_n) \) must be replaced by \( t_i \).’)

It is easy to see that the sentence logical rules for clause deduction together with this form of \((\forall E)\) give a complete deduction system for clauses without the identity symbol \( = \). A problem in connection with \((\forall E)\) in logic programming is to find a method which as effectively and economically as possible selects substitutions for testing. Letting the computer test substitutions by all the terms in the so-called Herbrand universe leads to an explosion in the number of variants which must be tried. No deduction except the most trivial should be performable by a
logic program within a reasonable time. The solution to this problem is Robinson's unification algorithm which we now describe.

4.4 Basic Ideas. The unification algorithm is based on two fundamental ideas.

(I) The purpose of substitutions in a deduction is to make it possible to apply resolution. (Later we will see that substitution also is needed to make contractions possible.) In the clauses

\[(1) \quad P(f(y),x) \leftarrow Q(x)\]
\[(2) \quad \leftarrow P(x,y), R(y)\]

it is impossible to apply the resolution rule since \(P(f(y),x)\) and \(P(x,y)\) are not identical. If we in (2) make the substitution \(\{x = f(y), y = z\}\), we get

\[(3) \quad \leftarrow P(f(y),x), R(x)\]

which can be resolved with (1):

\[(4) \quad \leftarrow Q(x), R(x)\]

2, Substitution

1,3, Resolution

We need therefore only test such substitutions which make resolution possible. All other possible substitutions may be excluded immediately by the algorithm.

(II) When we make a substitution, the substitution instance contains the same or less information than the premise, never more information. E.g., we have

\[\forall x \forall y P(x,y) \Leftarrow P(a,b)\]

\[P(a,b) \Leftarrow \forall x \forall y P(x,y)\]

i.e., \(P(a,b)\) contains less information than \(\forall x \forall y P(x,y)\).

When we substitute in order to unify two atomic formulas, we choose a substitution instance in such a way that we loose as little information as possible. We find the most general unifier. In the example above

\[(1) \quad P(f(y),x) \leftarrow Q(x)\]
\[(2) \quad \leftarrow P(x,y), R(y)\]

it should be possible to unify \(P(f(y),x)\) and \(P(x,y)\) by the substitution \(\{x = f(y), y = z\}\) in both (1) and (2):

\[(3) \quad P(f(y),y) \leftarrow Q(f(y))\]

1, Substitution

2, Substitution

Though \(\{x = f(y), y = z\}\) is a unifier for \(P(f(y),x)\) and \(P(x,y)\), it is not the most general unifier which can be seen by a comparison with the substitution we used earlier and which resulted in the clauses (1) and (3). The clauses (1), (3), (5) and (6) have the following standard forms:

\[(1) \quad \forall x \forall y \exists z \quad (Q(x) \rightarrow P(f(y),x))\]
\[(3) \quad \forall y \exists z \quad \neg(P(f(y),x) \land R(z))\]
\[(5) \quad \forall y \quad Q(f(y)) \rightarrow P(f(y),y)\]
\[(6) \quad \forall y \quad \neg(P(f(y),y) \land R(y))\]

We see that

\[(1) \Leftarrow (3)\]
\[(5) \Leftarrow (1)\]
\[(3) \Leftarrow (6)\]
\[(6) \Leftarrow (3)\]

Thus we loose more information by the substitution \(\{x = f(y), y = z\}\) than we do by the substitution we tested first.

The unification algorithm should find a substitution which makes resolution possible at the same time as the substitution should result in as small a loss of information as possible.

4.5 DEFINITION: Expression.

(1) A term is an expression.
(2) An atomic formula is an expression.
(3) A clause is an expression.

We use \(U, V, W\) in the metalanguage to designate expressions.

4.6 EXAMPLE. (I) The variables

\[x, y, z, x_1, x_2, \ldots\]

are expressions.

(II) The constants

\[a, b, c, 0, 1, 2, \pi, e\]

are expressions.
(III) The complex terms
\[ f(x); \ g(h(x),g(a,h(y))): \ (x + 0)(y + 1) \]
are expressions.

(IV) The atomic formulas
\[ P(x); \ Q(x,y); \ Q(f(x),f(a)): \ x + 0 \leq y; \ x+S(y) = x+y+x \]
are expressions.

(V) The clauses
\[ \leftarrow P(x), Q(x,y) \]
\[ P(f(x)) \leftarrow Q(x,y) \]
\[ Q(f(x),f(a)) \leftarrow \]
are expressions.

4.7 DEFINITION. (I) A substitution set
\[ \sigma = \{x_1 = t_1, \ldots, x_n = t_n\} \]
is a set of identity formulas
\[ x_i = t_i \]
where \( x_i \) is a variable and \( t_i \) is a term. It is assumed that no variable occurs as left-hand side in two distinct identities in a substitution set.

(II) If \( \sigma \) is a substitution set and \( U \) is an expression, then \( U \sigma \) is the result of substituting \( t_i \) for all occurrences of \( x_i \) in \( U \) (i = 1,...,n).

(*) The substitution set \( \sigma \) is a function according to the definition. This can be seen if we instead of \( x_i = t_i \) write \( (x_i,t_i) \). \( \sigma \) is then the one-place function \( \sigma = \{(x_1,t_1), \ldots, (x_n,t_n)\} \). \( \sigma \) is thus an operation, the substitution, which maps the expression \( U \) to a new expression \( \sigma(U) \) or, as it is traditionally written, \( U \sigma \). (*)

4.8 DEFINITION. The expression \( V \) is an instance of \( U \) iff there is a substitution \( \sigma \) such that
\[ V = U \sigma \]

4.9 EXAMPLE. (I) Let
\[ \sigma = \{x = f(y), y = a, z = g(v)\} \]
\[ \Theta = \{x = a, v = a\} \]

If \( U \) is the atomic formula
\[ P(h(x),y,v) \]
then \( U \Theta \) is the substitution instance
\[ P(h(f(y)),a,v) \]
and \( U \Theta \) is the instance
\[ P(h(a),y,a) \]
If \( V \) is the clause
\[ Q(x,y) \leftarrow P(x,y,z) \]
then \( V \Theta \) is the instance
\[ Q(f(y),a) \leftarrow P(f(y),a,g(v)) \]
while \( V \Theta \) is
\[ Q(x,y) \leftarrow P(x,y,z) \]
(II) Note that the substitution set may be empty, \( \sigma = \emptyset \). The empty substitution is designated by \( \epsilon \). We always have
\[ U \epsilon = U \]
Every expression is therefore an instance of itself.

4.10 DEFINITION. Let \( U \) and \( V \) be expressions. \( V \) is a variant of \( U \) iff \( V = U \sigma \) for some substitution \( \sigma = \{x_1 = y_1, \ldots, x_n = y_n\} \) where
(1) all variables in \( U \) occur among \( x_1, \ldots, x_n \)
(2) the \( y_i \) are pairwise distinct variables.

(*) We get \( V \) from \( U \) by changing variable names. (*)

4.11 EXAMPLE. The clause
\[ P(y,a) \leftarrow Q(x), R(f(x)) \]
is a variant of
\[ P(x,a) \leftarrow Q(y), R(f(x)) \]
(" let \( \sigma = \{x = y, y = x, z = z\} \) ")
and of
\[ P(x,a) \leftarrow Q(v), R(f(w)) \]
(" let \( \sigma = \{u = v, v = w, w = x\} \) ")
Note that the three clauses have the standard forms
∀y ∀x ∀z (Q(x) ∧ R(f(x))) → P(y,x)
∀x ∀y ∀z (Q(y) ∧ R(f(x))) → P(x,y)
∀u ∀v ∀w (Q(v) ∧ R(f(w))) → P(u,v)

They can be obtained from each other by permuting or changing bound variables. Therefore they are logically equivalent.

It is easy to show that if V is a variant of U, then U is a variant of V. Two clauses which are variants of each other are always logically equivalent. They therefore have the same content of information and the same logical properties.

4.12 DEFINITION. Let σ and θ be substitutions. The composition of σ and θ is the substitution σθ such that

\[ Uσθ = (Uσ)θ \]

for all expressions U

i.e., first σ is applied to U and then θ to Uσ.

(* Since substitutions are functions, compositions of substitutions are compositions of functions.*)

4.13 LEMMA. Let σ, δ, γ be substitutions. Then

1. \[ σε = εσ = σ \]
2. \[ (σδ)γ = σ(δγ) \]

4.14 EXAMPLE. Let

\[ U = P(x,f(y),z) \]
\[ σ = \{ x = f(y), y = z \} \]
\[ δ = \{ y = a, z = y \} \]

Then

\[ Uσθ = P(f(b),f(y),y) \]
\[ Uθσ = P(x,f(b),z) \]

Therefore composition of substitutions is not a commutative operation.

4.15 RULE FOR COMPOSITIONS. Let

\[ σ = \{ x_1 = t_1, \ldots, x_n = t_n \} \]
\[ θ = \{ y_1 = u_1, \ldots, y_m = u_m \} \]

Then

\[ σθ = \{ x_1 = t_1θ, \ldots, x_n = t_nθ, y_1 = u_1, \ldots, y_m = u_m \} \]
\[ = \{ x_i = t_iθ \} (t_iθ is x_i, 1 \leq i \leq n) \]
\[ = \{ y_j = u_j \} (y_j \in \{ x_1, \ldots, x_n \}, 1 \leq j \leq m) \]

4.16 DEFINITION. Let U and V be expressions.

(I) The substitution σ is a unifier for U and V iff \[ Uσ = Vσ. \]

(* A unifier makes two expressions identical.*)

(II) U and V are unifiable if they have a common unifier σ. To unify U and V is to apply a unifier σ to U and V.

(III) Let σ be a unifier for U and V. σ is the most general unifier (MGU) for U and V \[ ⇔ \] for every unifier θ for U and V there is a substitution γ such that \[ θ = σγ. \]

(* The unifier σ is an MGU for U and V if for every other unifier θ, it is true that Uθ is a substitution instance of Uσ. Every other unifier θ is a specialisation of σ, \[ θ = σγ. \] σ is thus at least as general as θ.*)

4.17 EXAMPLE. (I) The atomic formulas

\[ P(x,a) \text{ and } P(f(y),a) \]

are unifiable.

\[ σ = \{ x = f(b), y = b, z = a \} \]

is a unifier, though not an MGU, and gives \[ P(f(b),a). \]

\[ θ = \{ x = f(y), z = a \} \]

is an MGU and gives \[ P(f(y),a). \] We see that \[ P(f(b),a) \] is an instance of \[ P(f(y),a) \] but not the other way round.

(II) \[ P(x,a) \text{ and } P(f(y),b) \] are not unifiable because the constants a and b cannot be unified.

(III) \[ P(x,a) \text{ and } P(y,f(y)) \] are not unifiable because y and f(y) cannot be unified.
4.18 REMARK. (1) It can be shown that if the expressions \( U \) and \( V \) have a unifier, then they have an MGU.

II) It can be shown that if \( \sigma \) and \( \theta \) are two different MGU for \( U \) and \( V \), then \( U\sigma \) and \( U\theta \) are variants of each other (and consequently even \( V\sigma \) and \( V\theta \) are variants of each other). Therefore we sometimes use the expression the most general unifier.

III) The conditions for the unification of two expressions are the following.

1. Atomic formulae: Two atomic formulas can be unified if
   - the predicates have the same number of places, and
   - the predicates are identical, and
   - the arguments in the same place can be unified.

2. Terms: Two terms can be unified if
   - they are identical, or
   - one of them is a variable which does not occur in the other term, or
   - both terms are complex, and
     - they have the same number of places, and
     - they have the same initial function symbol, and
     - the arguments in the same place can be unified.

4.19 REMARK. When unification is used in a deduction, the purpose is almost always to be able to make resolution possible. We start from the clauses

\[
\begin{align*}
(1) & \quad P_1, \ldots, P_n, T_1 \leftarrow Q_1, \ldots, Q_m \\
(2) & \quad R_1, \ldots, R_p \leftarrow S_1, \ldots, S_q, T_2
\end{align*}
\]

and want to unify \( T_1 \) and \( T_2 \). The first step is to find an MGU \( \sigma \) for \( T_1 \) and \( T_2 \). Next the substitution \( \sigma \) is performed everywhere in the clauses (1) and (2):

\[
\begin{align*}
(3) & \quad P_1\sigma, \ldots, P_n\sigma, T_1\sigma \leftarrow Q_1\sigma, \ldots, Q_m\sigma & \text{1, Unification } \sigma \\
(4) & \quad R_1\sigma, \ldots, R_p\sigma \leftarrow S_1\sigma, \ldots, S_q\sigma, T_2\sigma & \text{2, Unification } \sigma
\end{align*}
\]

Since \( T_1\sigma \approx T_2\sigma \), we can eliminate \( T_1\sigma \) and \( T_2\sigma \) by resolution:

\[
\begin{align*}
(5) & \quad P_1\sigma, \ldots, P_n\sigma, R_1\sigma, \ldots, R_p\sigma \leftarrow Q_1\sigma, \ldots, Q_m\sigma, S_1\sigma, \ldots, S_q\sigma & \text{3, 4, Resolution}
\end{align*}
\]

Unifying in this way \( T_1 \) and \( T_2 \) in a deduction by an MGU such that the resolution operation becomes possible is also called to match \( T_1 \) and \( T_2 \).

4.20 EXAMPLE. (1) We start from

\[
\begin{align*}
(1) & \quad P(y,x) \leftarrow Q(x,y,x), \; x = f(y) \\
(2) & \quad \leftarrow P(x,a)
\end{align*}
\]

From Example 4.16, we know that \( \theta = \{ x = f(y), \; z = a \} \) is an MGU for \( P(y,x) \) and \( P(x,a) \).

\[
\begin{align*}
(3) & \quad P(f(y),x) \leftarrow Q(f(y),y,a), \; f(y) = f(y) & \text{2, Unification } \theta \\
(4) & \quad \leftarrow P(f(y),a) & \text{3, Unification } \theta \\
(5) & \quad \leftarrow Q(f(y),y,a), \; f(y) = f(y) & \text{3, 4, Resolution}
\end{align*}
\]

In the lines (3) and (4), we have matched \( P(f(y),x) \) with \( P(x,a) \) such that the resolution can be performed in line (5).

II) The unification algorithm in § 4.21 finds one common unifier \( \sigma \) for \( T_1 \) and \( T_2 \) and apply \( \sigma \) to the resolution premises (1) and (2) in the example of Remark 4.19. Doing resolution deductions by hand, it is often easier to find one substitution \( \sigma_1 \) which is applied to clause (1) and another substitution \( \sigma_2 \) which is applied to (2) such that \( T_1\sigma_1 = T_2\sigma_2 \).

E.g., we may unify \( P(f(y),x) \) and \( P(x,y) \) in

\[
\begin{align*}
(1) & \quad P(f(y),x) \leftarrow Q(x) \\
(2) & \quad \leftarrow P(x,y), \; R(y)
\end{align*}
\]

by using the substitutions \( \sigma_1 = x \) to (1) and \( \sigma_2 = \{ x = f(y), \; y = z \} \) to (2):

\[
\begin{align*}
(3) & \quad P(f(y),x) \leftarrow Q(x) & \text{1, Unification } \sigma_1 \\
(4) & \quad \leftarrow P(f(y),x), \; R(x) & \text{2, Unification } \sigma_2
\end{align*}
\]

This simpler procedure of matching two atomic formulas in two clauses by applying two substitutions will often be exploited in the present section.

4.21 The Unification Algorithm. We now give a formulation of the unification algorithm which determines an MGU for two terms or for two atomic formulas. Thus we presuppose that the expressions \( u \) and \( v \) to be unified are not clauses. The formulation is a modification of the algorithm in Genesereth and Nilsson, Logical Foundations of Artificial Intelligence.

Recursion Procedure \( \text{Unify}(u,v) \)

\[
\text{Begin} \quad u = v \implies \text{Return}(1) \\
\quad \text{Variable}(u) \implies \text{Return}(\text{NewVar}(u,v))
\]

194

195
Variable(v) -> Return(Mgu(var(v), u)),
Constant(u) or Constant(v) -> Return(False),
Not(Length(u) = Length(v)) -> Return(False),
Not(Part(u, 0) = Part(v, 0)) -> Return(False),
Begin i := 1,
σ := ( ),
Tag i > Length(u) -> Return(0),
σ := Mgu(Part(u, i), Part(v, i)),
σ = False -> Return(False),
σ := Compose(σ, σ),
u := Substitute(u, σ),
v := Substitute(v, σ),
i := i + 1,
Goto Tag
End
End

Procedure Mgu(var(u, v)
Begin
Not Term(v) -> Return(False),
Occur(var(u, v)) -> Return(False),
Return((u = v))
End

If the two expressions u and v are unifiable, the procedure gives an MGU. If u and v are not unifiable, the algorithm answers False.

There are several undefined components in the procedure:
Variable(u) is true <=> u is a variable
Constant(u) is true <=> u is a constant
Term(u) is true <=> u is a term
Occur(var(v) is true <=> u is a variable which occurs in the term v
Length(P(t_1 ... t_n)) = the number of places in P = n
Length(H(t_1 ... t_n)) = the number of places in f = n
Part(P(t_1 ... t_n), 0) = P
Part(P(t_1 ... t_n), i) = t_i for 1 ≤ i ≤ n
Part(H(t_1 ... t_n), i) = f
Part(H(t_1 ... t_n), i) = t_i for 1 ≤ i ≤ n
Compose(σ, θ) = the composition σθ of the substitutions σ and θ
Substitute(u, σ) = the result of applying the substitution σ to the expression u.

When we want to match two atomic formulas in a deduction, we may use the unification algorithm or we may directly use the definition of MGU. The unification algorithm is built into all implementations of Prolog. The system therefore automatically performs the unifications in the deductions.

We now explain by a couple of examples how the unification algorithm works.

4.22 EXAMPLE. Use the unification algorithm to unify the atomic formulas P(t_1, t_2, t_3) and P(u_1, u_2, u_3) which we assume are unifiable.

SOLUTION:
First the algorithm examines whether the formulas are identical. If they are identical, the algorithm answers σ = ε = { }. We now assume that they are different, i.e., t_i and u_i are different for at least one i. The algorithm examines whether the formulas are of the same length. We get

Length(P(t_1, t_2, t_3)) = Length(P(u_1, u_2, u_3)) = 3

The algorithm examines whether the formulas have the same predicate. Since

Part(P(t_1, t_2, t_3), 0) = Part(P(u_1, u_2, u_3), 0) = P

even this condition for unification is satisfied.

The algorithm computes Mgu(t_1, u_1) = σ_1 and makes the substitutions

P(t_1 σ_1 t_2 t_3) 
P(u_1 σ_1 t_2 t_3)

Next t_2 σ_1 and t_2 σ_1 are unified

Mgu(t_2 σ_1, u_2 σ_1) = σ_2

and this substitution is applied

P(t_1 σ_1 σ_2 σ_3 t_2 σ_3) 
P(u_1 σ_1 σ_2 σ_3)

Finally, t_3 σ_1 σ_2 and t_3 σ_1 σ_2 are unified

Mgu(t_3 σ_1 σ_2 σ_3, u_3 σ_1 σ_2) = σ_3

The algorithm puts

σ := σ_1 σ_2 σ_3
and makes the substitution
\[ P(t_1 \sigma_1 t_2 \sigma_3 t_4 \sigma_5 t_6 \sigma_7 t_8) \]
\[ P(u_1 \sigma_1 t_2 \sigma_3 u_2 t_4 \sigma_5 u_3 t_6 \sigma_7 u_4 t_8) \]
The algorithm sets \( i := 4 \). Since we now have the index \( i = 4 > 3 = \text{Length}(P(t_1 \sigma_1 t_2 \sigma_3 t_4 \sigma_5 t_6 \sigma_7 t_8)) \), the algorithm halts and gives as the answer the following MGIU for \( P(t_1 \sigma_1 t_2 t_3) \) and \( P(u_1 u_2 u_3) \):
\[ \sigma = \sigma_2 \sigma_3 \]

The idea is very simple. First \( t_1 \) and \( u_2 \) are unified by \( \sigma_1 \). Next \( \sigma_1 \) is specialised exactly so much that we even get a unification of \( t_2 \) and \( u_2 \). This yields the substitution \( \sigma_2 \sigma_3 \). Finally, \( \sigma_1 \sigma_2 \) is specialised further so that also \( t_3 \) and \( u_3 \) are unified by \( \sigma_1 \sigma_2 \sigma_3 \). Then unifies \( P(t_1 \sigma_1 t_2 t_3) \) and \( P(u_1 u_2 u_3) \).

4.23 EXAMPLE. Unify the terms
\[ f(x, g(y, x)) \] and \[ f(x, x) \]

**SOLUTION:**
The terms have the same length 2 and the same function symbol \( f \). First we unify the arguments \( x \) and \( x \). Since they are identical, the algorithm gives
\[ \text{Mgu}(x, x) = \varepsilon = \{ \} \]

We determine \( \text{Mgu}(g(y, x), x) \). Since \( x \) is a variable, we get
\[ \text{Mgu}(g(y, x), x) = \text{Mguvar}(x, g(y, x)) \]

The procedure \( \text{Mguvar} \) gives
\[ \text{Mgu}(g(y, x), x) = \text{Mguvar}(x, g(y, x)) = \{ x = g(y, x) \} \]
The unifier for \( f(x, g(y, x)) \) and \( f(x, x) \) is therefore the composition
\[ \sigma = (x = g(y, x)) \]

The result of the unification is
\[ f(x, g(y, x)) \sigma = f(g(y, x), g(y, x)) \]
\[ f(x, x) \sigma = f(g(y, x), g(y, x)) \]

4.24 REMARK. (I) According to Definition 4.16, \( y \) and \( f(y) \), e.g., cannot be unified. Every substitution \( \sigma = \{ y = \ldots \} \) yields two different

terms \( t \) and \( f(t) \). Generally, it is true that if \( U \) is a variable and \( V \) a complex term in which \( U \) occurs, then \( U \) and \( V \) cannot be unified. The line

\[ \text{Occur}(u, v) \implies \text{Return}(\text{False}) \]

in \( \text{Mguvar} \) blocks the unification of \( U \) and \( V \). This part of the algorithm is called the **occurrence check**.

(II) The occurrence check is necessary in connection with unifications in deductions because otherwise it is possible to deduce \( u \rightarrow v \) from consistent sets of premises. But the occurrence check also complicates the unification part of deductions. If we in the premises

\[
\begin{align*}
(1) & \quad P(y) \leftarrow \\
(2) & \quad u \leftarrow P(f(y))
\end{align*}
\]

try directly to use the unification algorithm to unify \( P(y) \) and \( P(f(y)) \), we see that this does not work. The occurrence check blocks the unification of \( y \) and \( f(y) \). We are therefore unable to deduce \( u \leftarrow v \). If we pass to the standard form, we see that the premises (1) and (2) are inconsistent.

\[
\begin{align*}
(1) & \quad \forall y \ P(y) \\
(2) & \quad \forall y \leftarrow P(f(y)) \\
(3) & \quad \neg P(f(y)) \\
(4) & \quad P(f(y)) \\
(5) & \quad \bot
\end{align*}
\]

The solution to this problem is to take care that two clauses, to which we want to apply the unification algorithm, do not contain any common variables. Then variables should be changed in one or both of the clauses so that they are without common variables. This is possible since all variables in a clause are bound. The bound variables are changed in accordance with the equivalence

\[ \forall x \ A(x) = \forall y \ A(y) \quad \text{if } y \text{ is a new variable not occurring in } A(x) \]

Using variable change, we get the following deduction in clause form:

\[
\begin{align*}
(1) & \quad P(y) \leftarrow \\
(2) & \quad u \leftarrow P(f(y)) \\
(3) & \quad P(y_2) \leftarrow \\
(4) & \quad P(f(y)) \\
(5) & \quad P(f(y)) \leftarrow \\
(6) & \quad \bot
\end{align*}
\]

1. variable change
2. Unification
3. Unification
4. 5. Resolution
4.25 EXAMPLE. Show by resolution and unification

\[ \exists x \, P(x), \, \forall x \, (P(x) \rightarrow Q(x)) \rightarrow \exists x \, (P(x) \land Q(x)) \]

SOLUTION:
We write the premises in clause form.

(1) \( P(a) \leftarrow \)
(2) \( Q(a) \leftarrow P(x) \)

The negation of the conclusion

\[ \neg \exists x \, (P(x) \land Q(x)) \]

is equivalent with

\[ \forall x \, \neg \lnot (P(x) \land Q(x)) \]

which has the clause form

(3) \( \neg P(x), \neg Q(x) \)

We now get the following resolution proof:

(1) \( P(a) \leftarrow \)
(2) \( Q(x) \leftarrow P(x) \)
(3) \( \neg P(x), \neg Q(x) \)
(4) \( \leftarrow P(x), P(x) \)
(5) \( P(x) \leftarrow \)
(6) \( \leftarrow P(x), P(x) \)
(7) \( \leftarrow P(x) \)
(8) \( \leftarrow \)

4.26 EXAMPLE. In this example, we study relations in the British Royal Family. We use the following constants and predicates.

c = Queen Elizabeth
p = Prince Philip
c = Prince Charles
d = Princess Diana
w = Prince William (* w is here used as constant, not as a variable.*)
F(x,y): x is father of y
M(x,y): x is mother of y
P(x,y): x is paternal grandparent of y

We have the premisses

\( F(p,c): Philip \) is father of Charles
\( M(e,c): Elizabeth \) is mother of Charles
\( F(c,w): Charles \) is father of William
\( M(d,w): Diana \) is mother of William
together with the premiss

\[ \forall x \, \forall y \, (\exists z \, (F(x,z) \lor M(x,z)) \land F(x,y)) \rightarrow P(x,y) \]

i.e., x is a paternal grandparent of y if there is some z which is the father of y and has x as his father or mother.

We now ask if William has paternal grandparents. We do this by assuming that William has no paternal grandparents

\[ \neg \exists x \, P(x,w) \]

If this assumption together with the other premisses leads to a contradiction, then William must have at least one paternal grandparent. We now carry out a clause logical deduction of \( \leftarrow \) from the premisses.

(1) \( F(p,c) \leftarrow \)
(2) \( M(e,c) \leftarrow \)
(3) \( F(c,w) \leftarrow \)
(4) \( M(d,w) \leftarrow \)
(5) \( P(x,y) \leftarrow F(x,z), F(z,y) \)
(6) \( P(x,y) \leftarrow M(x,z), F(z,y) \)
(7) \( \leftarrow F(x,w) \)
(8) \( \leftarrow F(x,w), F(z,w) \)
(9) \( \leftarrow F(x,z), F(z,w) \)
(10) \( \leftarrow F(x,z), F(z,w) \)
(11) \( \leftarrow F(x,w) \)
(12) \( \leftarrow \)

(* In the lines (8), (10), (11), the atomic formula is underlined which is resolved away in the following line.*)

Thus the deduction answers \( \neg \exists z \) to the question whether William has paternal grandparents. Since it is the substitution in line (10)

\[ x = p = Philip \]

for the variable x in our question

(7) \( \leftarrow P(x,w) \)

which leads to \( \leftarrow \) in the deduction, our deduction also gives the answer
Philip is a paternal grandparent of William.  
There is another deduction having the same premises which gives the answer  
Elizabeth is a paternal grandparent of William.

4.27 REMARK. In the literature on logic programming, it is more common to write the deduction in the preceding example in the following form. The premises (1)-(6) are given first, as program sentences.

(P1) \( F(p,z) \leftarrow \)
(P2) \( M(x,z) \leftarrow \)
(P3) \( F(c,w) \leftarrow \)
(P4) \( M(d,w) \leftarrow \)
(P5) \( P(x,y) \leftarrow P(x,z), P(z,y) \)
(P6) \( P(x,y) \leftarrow M(x,z), P(z,y) \)

The question \( \leftarrow P(x,w) \) is then taken as the starting point of the deduction which is written in the following form:

\( \leftarrow P(x,w) \)
\((^* \text{ (P5): } y = w ^*)\)
\( \leftarrow P(x,y), P(x,w) \)
\((^* \text{ (P1): } x = p, x = c ^*)\)
\( \leftarrow P(x,y) \)
\((^* \text{ (P3) } ^*)\)
\( \leftarrow \)

The advantage of this way of writing is that the linear form of the deduction as a backward deduction from the question to the contradiction \( \leftarrow \) becomes clear. The comments state with which program sentence we are resolving and what substitution is used at the unification.

Note that this simple linear form only turns up when we limit ourselves to Horn classes. If we have classes with alternatives consequents, the deductions get a more complicated structure. That is one more reason why we limit ourselves to Horn classes in Prolog.

4.28 Search Spaces. In the figure, the search space for the problem studied in Example 4.26-27 of giving a backward solution to the problem \( \leftarrow P(x,w) \) from the six program sentences (P1)-(P6). The six different branches give all possible ways of trying a backward deduction.

Two of the branches, indicated by STOP, cannot lead to \( \leftarrow \). The reason is that the deductions have reached a point where it is impossible to go on and resolve further. The first branch from the left is the deduction of \( \text{§ 4.27} \). The branches 1 and 3 from the left give the answer  
Philip is a paternal grandparent of William.

The branches 4 and 6 give the answer  
Elizabeth is a paternal grandparent of William.

One searches for a backward deduction for the given problem by searching through the search space. There are two main strategies to follow in the search: breadth-first and depth-first.

In a breadth-first search, all deductions are first developed one step. This gives the vertex of the tree:

\( \leftarrow P(x,w) \)

Next, all deductions are developed parallel to step two. This develops the tree into two ramifications:

\( \leftarrow P(x,w) \)
\( \leftarrow P(x,z), P(z,w) \)

Next, all deductions are developed parallel to step 3. It develops the tree to six ramifications in our example. In this way one continues until a deduction of \( \leftarrow \) turns up or until the search space has been searched through.

In a depth-first search, the leftmost branch is examined first until its end point. If the branch ends in \( \leftarrow \), the search stops. Then we have found a deduction. Otherwise, one backs in the tree and examines the second branch from the left. In this way, the search through the branches from left to right goes on until a deduction of \( \leftarrow \) is found, or the search tree has been completely examined from left to right without a deduction of \( \leftarrow \) having turned up.

The advantage of the breadth-first method is that it is complete; if there is a deduction for the problem, the method will also find one. The drawback is that the method is slow.
The depth-first method is mostly faster than breadth-first; but it is not complete. There are solvable deduction problems whose search spaces have one or more infinite branches. The depth-first method may get lost in such an infinite branch and never succeeds in entering a branch containing a deduction of ← though there are such branches in the tree. An example is given in Section 5.

4.29 EXAMPLE. There are deduction problems which cannot be solved solely by resolution and unification. The clauses

(1) \( P(x), P(y) \leftarrow \)
(2) \( \leftarrow P(x), P(v) \)

are inconsistent with each other. This can be seen by changing to standard form.

\[
\begin{align*}
(1) & \quad \forall x \forall y (P(x) \lor P(y)) \\
(2) & \quad \forall x \forall y \neg(P(x) \land P(v)) \\
(3) & \quad P(x) \lor P(x) \\
(4) & \quad \neg(P(x) \land P(x)) \\
(5) & \quad \neg P(x) \\
(6) & \quad P(x) \\
(7) & \quad \bot \\
(8) & \quad P(x) \\
(9) & \quad P(x) \land P(x) \\
(10) & \quad \bot
\end{align*}
\]

But all attempts of applying unification and resolution to the clauses (1) and (2) yield resolvents with two atomic formulas. We will never reach the clause \( \leftarrow \) which contains zero atomic formulas.

A proof in clause form where we use a new deduction rule, the contraction rule, may look as follows.

\[
\begin{align*}
(1) & \quad P(x), P(y) \leftarrow \\
(2) & \quad \leftarrow P(x), P(v) \\
(3) & \quad P(x), P(x) \leftarrow \\
(4) & \quad P(x) \leftarrow \\
(5) & \quad \leftarrow P(x), P(v) \\
(6) & \quad \leftarrow P(v)
\end{align*}
\]

(* We have now unified \( P(x) \) and \( P(y) \) in the consequent of Premiss (1).*)
\begin{align*}
(7) & \leftarrow P(x) \\
(8) & \leftarrow \\
\text{First } P(x) \text{ and } P(y) \text{ in the consequent of}
\end{align*}
\begin{align*}
(1) & \quad P(x), P(y) \leftarrow \\
(3) & \quad P(x), P(x) \leftarrow \\
\text{which has the standard form}
\end{align*}
\begin{align*}
& \hspace{1cm} P(x) \lor P(x)
\end{align*}
\begin{align*}
\text{The sentence logical equivalence}
\end{align*}
\begin{align*}
A \lor A \Leftrightarrow A
\end{align*}
\begin{align*}
\text{yields}
\end{align*}
\begin{align*}
P(x)
\end{align*}
\begin{align*}
\text{with the clause form}
\end{align*}
\begin{align*}
(4) & \quad P(x) \leftarrow \\
\text{To perform a contraction thus consists in drawing together } A \lor A \text{ to } A.
\end{align*}

\textbf{4.30 The Contraction Rule. Applying the contraction rule, also called the factoring rule, consists of two steps.}

\textit{(I) Predicate logical step:} Let a clause be given
\begin{align*}
(1) & \quad P_1, \ldots, P_m, Q, R \leftarrow S_1, \ldots, S_n
\end{align*}
where \(Q\) and \(R\) are unifyable. Let \(\sigma\) be an MGU for \(Q\) and \(R\) such that
\begin{align*}
Q\sigma = R\sigma = T
\end{align*}
Apply \(\sigma\) to the clause \((1)\):
\begin{align*}
(2) & \quad P_1\sigma, \ldots, P_m\sigma, T, T \leftarrow S_1\sigma, \ldots, S_n\sigma
\end{align*}
\textit{(II) Sentence logical step:} Since
\begin{align*}
T \lor T \Leftrightarrow T
\end{align*}
we may eliminate one of the \(T\)'s.
\begin{align*}
(3) & \quad P_1\sigma, \ldots, P_m\sigma, T \leftarrow S_1\sigma, \ldots, S_n\sigma
\end{align*}
This completes the contraction operation.

\textbf{* The contraction rule allows only contractions in the consequent of a clause. The completeness theorem 6.9 in Section 6 shows that we do not in addition need a rule which allows contractions in the antecedent. *}

\textbf{4.31 EXAMPLE.} The barber paradox is about the barber in the small village of Nowhere.

\begin{enumerate}
\item The barber is a man and he lives in Nowhere.
\item The barber shaves those men in Nowhere and only those men in Nowhere who do not shave themselves.
\end{enumerate}

\textbf{Symbols:}
\begin{itemize}
\item \(b\): the barber
\item \(M(x)\): x is a man
\item \(N(x)\): x lives in Nowhere
\item \(S(x,y)\): x shaves y
\end{itemize}
Then (1) and (2) may be formalised in standard form
\begin{align*}
(1) & \quad M(b) \land N(b) \\
(2) & \quad \forall x (M(x) \land N(x) \rightarrow (R(b,x) \leftrightarrow \neg R(x,x)))
\end{align*}
The sentences (1) and (2) can be shown to be inconsistent with each other by changing to clause form and deducing \(\leftarrow\). It is necessary to use contraction in the deduction.

\textbf{4.32 REMARK. (I) All valid deductions can be performed in clause form by using only}
\begin{itemize}
\item resolution
\item unification
\item contraction
\end{itemize}
\textbf{(II) If we limit ourselves to Horn clauses, all valid deductions can be performed using only}
\begin{itemize}
\item resolution
\item unification
\end{itemize}
This simplification is still another reason why only Horn clauses are used in Prolog.

\textbf{4.33 Identity.} In the deduction system PD for predicate logic in standard form we have special deduction rules (Refl) and (Subst) for identity. In connection with deduction in clause form, we only have the de-
4.34 **LEMMA.** Identity in clause logic is reflexive, symmetric, and transitive:

(1) \( x = x \leftarrow \)

(2) \( x = y \leftarrow y = x \)

(3) \( x = z \leftarrow x = y, y = z \)

4.35 **EXAMPLE.** Show

\[ \forall x \neg R(x,x) \leftarrow \forall x \forall y (R(x,y) \rightarrow x \neq y) \]

**SOLUTION:**

(i) **Clause form:**

- The premise in clause form:
  
  (A1) \( x + 0 = x \leftarrow \)
  
  (A2) \( x + S(y) = S(x + y) \leftarrow \)

- Furthermore we need the following identity axioms according to § 4.33.

  (B1) \( x = x \leftarrow \)
  
  (B2) \( S(x) = S(y) \leftarrow x = y \)
  
  (B3) \( x_1 + y_1 = y_1 \leftarrow x_1 = y_1, y_1 \neq y_2 \)
  
  (B4) \( x_1 + y_2 = y_1 \leftarrow y_2, y_1 \neq x_1 \)

where \( S \) is the successor function. The equations give rise to the program sentences

\[ x = x \leftarrow \]

\[ S(x) = S(y) \leftarrow x = y \]

\[ x_1 + y_1 = y_1 \leftarrow x_1 = y_1, y_1 \neq y_2 \]

\[ x_1 + y_2 = y_1 \leftarrow y_2, y_1 \neq x_1 \]
(D1) \[ 1 = S(0) \leftarrow \]

(D2) \[ 2 = S(1) \leftarrow \]

(D3) \[ 3 = S(2) \leftarrow \]

(D4) \[ 4 = S(3) \leftarrow \]

To compute \( 2 + 2 \), we ask whether there is a number \( z \) such that 
\[ 2 + 2 = z. \]
We ask the question by assuming that there is no such \( z \), i.e.,
\[ \leftarrow 2 + 2 \neq z \]
and derive \( \leftarrow \) from this assumption. By noting which substitution for \( z \)
it is that leads to \( \leftarrow \), we obtain the result of the addition \( 2 + 2 \).

**Deduction:**

1. \( \leftarrow 2 + 2 \neq z \)
2. \( 2 + 2 = 2 + S(1) \leftarrow 2 = 2, 2 = S(1) \) (E3), Unification
3. \( 2 + 2 = 2 + S(1) \leftarrow 2 = 2 \) 2, (D2), Resolution
4. \( 2 = 2 \leftarrow \) (E1), Unification
5. \( 2 + 2 = 2 + S(1) \leftarrow 3, 4, Resolution \)
6. \( 2 + 2 = z \leftarrow 2 + S(1) = z, 2 + 2 = 2 + S(1), z = z \) (E4), Unification
7. \( \leftarrow 2 + S(1) = z, 2 + z = 2 + S(1), z = z \) 1, 6, Resolution
8. \( \leftarrow z = 2 + S(1) \leftarrow 5, 7, Resolution \)
9. \( z = z \leftarrow \) (E1), Unification
10. \( \leftarrow 2 + S(1) = z \) 8, 9, Resolution

\(*\) (1) is now in (10) transformed such that (A2) can be applied.*

11. \( \leftarrow 2 + 2 = S(2 + 1) \leftarrow \) (A2), Unification
12. \( 2 + 2 = S(2 + 1) \leftarrow z = 2 + S(1) = S(2 + 1), z = z \) (E4), Unification
13. \( \leftarrow S(2 + 1) = z, 2 + S(1) = S(2 + 1), z = z \) 10, 12, Resolution
14. \( \leftarrow S(2 + 1) = z, z = z \) 11, 13, Resolution
15. \( \leftarrow S(2 + 1) = z \) 9, 14, Resolution

\(*\) We now compute \( 2 + 1 \) by showing \( 2 + 1 = 2 + S(0) = S(2 + 0) = S(2) = 3 \).

16. \( \leftarrow 2 + 1 = 2 + S(0) \leftarrow 2 = 2, 1 = S(0) \) (E3), Unification
17. \( \leftarrow 2 + 1 = 2 + S(0) \leftarrow 2 = 2 \) 16, (D1), Resolution
18. \( \leftarrow 2 + 1 = 2 + S(0) \leftarrow \) 4, 17, Resolution
19. \( \leftarrow 2 + S(0) = S(2 + 0) \leftarrow \) (A2), Unification
20. \( \leftarrow 2 + 1 = S(2 + 0) \leftarrow 2 + S(0) = S(2 + 0), 2 + 1 = 2 + S(0), S(2 + 0) = S(2 + 0) \) (E4), Unification
21. \( \leftarrow 2 + 1 = S(2 + 0) \leftarrow 2 + 1 = 2 + S(0), S(2 + 0) = S(2 + 0) \) 

Since it was the substitution \( z = 4 \) which resulted in the deduction of \( \leftarrow \), the answer is that \( 2 + 2 = 4 \).

4.37 REMARK. The example is partly unrealistic. There are faster algorithms for addition which can be represented and applied even in Horn clauses. But the example shows in principle how computation algorithms, and other algorithms as well, can be applied in logic programming. Applying an algorithm to a problem consists in a step-by-step de-
duction of the solution from the basic principles of the algorithm and a formulation of the given problem. In the example, the computation is made by a deduction in clause logic of the result $2 + 2 = 4$ from the basic principles of addition expressed in (A1) and (A2).

4.38 CAUTION. (I) If we want to deduce

$$A_1, \ldots, A_n \vdash B$$

where B is not a conjunction, we always, in agreement with Method 3.10, apply indirect deduction:

(1) We write $A_1, \ldots, A_n$ in clause form and let them belong to the set of premises.

(2) We write $\neg B$ in clause form and let the clause or clauses belong to the set of premises.

(3) We deduce $\vdash$.

(II) It might be tempting to use a combination of hypothetical and indirect deduction when we try to deduce a conclusion which is an implication. If, e.g., we are going to deduce the conclusion

$$\forall x (P(x) \rightarrow Q(x))$$

it might be tempting to do as in a predicate logical deduction in PD and assume $P(x)$ and $\neg Q(x)$ as extra premises, i.e., add as extra premises the clauses

$$P(x) \vdash$$

$$\neg Q(x)$$

Such a procedure results, however, in error. E.g., it is possible to "prove" by such a hypothetical deduction that

$$\forall x P(x) \rightarrow \exists x Q(x) \vdash \forall x (P(x) \rightarrow Q(x))$$

First we write the premise in Skolem form:

$$\forall x P(x) \rightarrow \exists x Q(x) \iff \exists x \exists y (P(x) \rightarrow Q(y))$$

which gives the Skolem form

$$P(a) \rightarrow Q(b)$$

and the clause

(1) $Q(b) \vdash P(x)$

For hypothetical deduction, we assume

(2) $P(x) \vdash$

For indirect deduction, we assume the negation of the consequent

(3) $\neg Q(x) \vdash$

We now get the following "clause proof":

(1) $Q(b) \vdash P(x)$
(2) $P(x) \vdash$
(3) $\neg Q(x) \vdash$
(4) $\neg Q(b) \vdash 3, \text{ Unification}$
(5) $\neg P(x) \vdash 1, 4, \text{ Resolution}$
(6) $P(a) \vdash 2, \text{ Unification}$
(7) $\vdash 5, 6, \text{ Resolution}$

But the model $\mathcal{M} = (\mathcal{M}, P, Q) = (\{a, b\}, \{a\}, \{b\})$ shows that

$$\forall x P(x) \rightarrow \exists x Q(x) \in \forall x (P(x) \rightarrow Q(x))$$

From the example we can see why hypothetical deduction does not work for sentences in clause form. The premises (2) and (3) have the standard forms:

(2) $\forall x P(x)$
(3) $\forall x \neg Q(x)$

They are much stronger than their counterparts used by hypothetical deduction in the standard form.

Use always consistently and throughout indirect deduction when making deductions in clause form. Avoid hypothetical deduction.

4.39 Heuristic Rule (Horn clauses). From a number of premises in Horn clause form we want to deduce $\vdash$.

(1) Among the premises, at least one clause with an empty consequent must occur

$$\vdash P_1, \ldots, P_n$$

because otherwise it is impossible to deduce $\vdash$. Take such a clause as the starting point.

(2) Find a premise

$$Q \leftarrow R_1, \ldots, R_m$$
whose consequent Q can be matched with one of the P_i, e.g., P_1. Find an
MGU & for P_1 and Q and apply it to the two clauses. Eliminate by
resolution P_1 & and Q&:

\[ \text{\textless P_1} \text{\&} \rightarrow \text{R_1} \text{\&} \rightarrow \text{R_n} \text{\&} \]

(3) Find a premise

\[ S \leftarrow \text{T_1}, \ldots, \text{T_n} \]

such that S can be matched with some P_1 & or R_j &. Unify and resolve
again.

(4) All the time, the clause with empty consequent which we obtained in
the preceding line is in the next step resolved with one of the premises
until we finally arrive at \( \text{\textless} \). We use exclusively backward
resolution.

4.40 REMARK. In deductions where premises occur which are not of
Horn type it does not suffice with only backward resolution. At least one
time, one must use resolution in its general form as defined in § 3.2. In
addition, it may in rare cases be necessary to perform contraction.

9-5 The Road to Prolog

5.1 The idea of using predicate logic as a programming language is simple
and natural as the following considerations show.

(I) User Closeness. A programming language is a formal language.
Predicate logic is presumably the user closest of all formal languages
in the sense that predicate logic is closer to the natural languages than
other formal languages are. The clause form is admittedly less user close
than the standard form but still in this respect superior to other formal
languages.

(II) Data Bases. An important application of programs is the handling
of data bases. A set of information is stored in a data base. We want to get
out the information which is present explicitly or implicitly in the data
base. The explicit information is represented in the data base in a
form which makes it immediately recognisable. An efficient search routine
may find such information (see Item (III) below). To get out information
which is implicit in the data base, some data processing is needed.

Predicate logic is the ideal instrument for this sort of data processing. A
deduction of

\[ \text{A_1}, \ldots, \text{A_n} \leftarrow \text{B} \]

makes the piece of information B explicit which is otherwise only
implicitly present in the sentences A_1, ..., A_n. If the sentences A_1, ..., A_n
express information which occurs explicitly in the data base, the
deduction has taken out and made explicit even the piece of information B
which before only existed implicitly in the data base. As soon as the
piece of information has been made explicit, it can be communicated to
and applied by a user.

A simple example of such an application was given in Example 4.26. The data base contains information about who is father or
mother of whom in the British Royal Family. In addition, a definition of the
paternal grandparent relation is included. Implicit in these sentences
is information about who are paternal grandparents of prince William.
Predicate logic can find this piece of information, extract it and make it
explicit as was shown in Example 4.26.

(III) Algorithms. Another important application of programming lan-
guages is the formulation and execution of algorithms. Information may
be expressed as sentences in the language of predicate logic. This is also
true of information on the basic principles of a given algorithm. All
algorithms may be formulated and executed in predicate logic. Executing
an algorithm is the same as deducing from these basic principles a solution
to the problem to which the algorithm is applied.

The algorithm may, e.g., be an administrative routine or a com-
putation algorithm. In § 4.36, an example involving a computation algo-
rithm was given. The basic principles for addition are expressed in the
predicate logical sentences (A1) and (A2). From the sentences (A1) and
(A2) (and some auxiliary premises) we deduced the solution \( 2 + 2 = 4 \)
to the computation problem "What is 2 + 2?"

5.2 Prolog. Predicate logic in standard form with a deduction system
like, e.g., PD in Hansen (1992, 1994) can in principle be used as a pro-
gramming language. Programs in this language are, however, much too
slow. In Section 2-7, a transformation of predicate logic with the fol-
lowing constituents is described:

(1) Transition to clause form.
(2) Limitation to Horn clauses.
(3) Application of resolution, in particular backward resolution.
(4) Application of unification.

This transformation makes the search for solutions to deduction problems so much more efficient than it becomes possible to use predicate logic as a programming language, even in practice. (Strictly speaking, it is only a fragment of predicate logic, Horn sentences or Horn clauses, which is used.) Prolog (i.e., PROgramming in LOGic) is a programming language which is based on predicate logic in this way.

Logic programming is the use of predicate logic as a programming language where the program is executed by a deduction. Prolog is logic programming in Horn clauses with resolution and unification as the only deduction rules.

5.3 Deductions in Prolog. A Prolog program consists of a number of program sentences of the form

\[ P \leftarrow Q_1, \ldots, Q_n \]

They all belong to the set of premises of the Prolog deduction. In addition, there is a sentence of the form

\[ \leftarrow R_1, \ldots, R_m \]

in the set of premises. It is sometimes called the question or the goal. We want to know if \( R_1 \wedge \ldots \wedge R_m \) follows from the program sentences. We therefore assume \( \neg (R_1 \wedge \ldots \wedge R_m) \), i.e., \( \leftarrow R_1, \ldots, R_m \), as an extra premise and feed it into the system. If Prolog is able to deduce \( \leftarrow \) from the premises, the answer is 'Yes, \( R_1 \wedge \ldots \wedge R_m \) is true'. If Prolog is not able to deduce \( \leftarrow \) from the premises, the answer is 'No, \( R_1 \wedge \ldots \wedge R_m \) is not true'.

Prolog applies only backward deduction. A Prolog deduction has, with one simplification, the same structure as the one indicated in § 4.39 for deductions in Horn clauses. Since, in Prolog deductions, it is always the question which is negative, i.e., has the form

\[ \leftarrow R_1, \ldots, R_m \]

the question is taken as starting point in the deduction. The deduction therefore becomes a linear backward deduction from the question to \( \leftarrow \). Every sentence in the deduction, from the question to the second to last sentence, is resolved in the following line with a program sentence.

5.4 Rules for the Search Order. A Prolog program consists of a sequence of program sentences

\[
\begin{align*}
(P1) & \quad Q \leftarrow P_1, \ldots, P_n \\
(P2) & \quad S \leftarrow R_1, \ldots, R_m \\
& \quad \ldots \ldots \ldots \\
\end{align*}
\]

The deduction consists of a sequence of classes having the form

\[ \leftarrow T_1, \ldots, T_k \]

Each such clause, except the last one, is resolved with one of the program sentences. This is done by matching one of the \( T_i \) with the consequent in a program sentence and then resolving the program sentence and the clause \( \leftarrow T_1, \ldots, T_k \) with each other. A rule is needed to determine the order in which the \( T_i \) are tested for matching. For each choice of \( T_i \), a rule is needed to determine the order in which the program sentences (P1) are tested for matching with \( T_i \) and possibly resolution. Several different such rules occur in different implementations of Prolog. A simple and quite common solution is the following.

- The atomic formulas \( T_i \) are tested from left to right.
- For each choice of \( T_i \), the program sentences are tested in the order they are given, i.e., from the top downwards.

5.5 EXAMPLE. The problem is defined by the following program sentences:

\[
\begin{align*}
(P1) & \quad P(x) \leftarrow Q(x), R(x,y) \\
(P2) & \quad R(x,y) \leftarrow S(x) \\
(P3) & \quad Q(x) \leftarrow \\
(P4) & \quad S(x) \leftarrow \\
& \quad \quad \quad \quad \leftarrow Q(a), P(a) \\
& \quad \quad \quad \quad \quad \text{Question} \\
& \quad \quad \quad \quad \quad (* \text{According to the rule, } Q(a) \text{ should be tested first. } Q(a) \text{ cannot be matched with } P(x) \text{ in (P1) or } R(x,y) \text{ in (P2). } Q(a) \text{ can be matched with } Q(x) \text{ in (P3).} \\
& \quad \quad \quad \quad \quad (* \text{ We test from the top in (P1)-(P4) whether } P(a) \text{ can be matched with some consequent. By the rule, we must choose the first which will do, i.e., (P1).} \\
\end{align*}
\]
(II) A deduction may be finite and unsuccessful. Then it ends in a clause which cannot be matched and resolved with any program sentence. If Prolog gets into such a branch, the program elastuates the deduction as far as possible and then continues the search for a deduction of $\leftarrow$ in another branch. The branches 2 and 5 in the tree in § 4.28 represent such deductions.

(III) A deduction may be infinite. It is unsuccessful since it does not end in $\leftarrow$. The deduction problem

$$ P(x) \leftarrow P(y) $$

$$ P(x) $$

has, e.g., only one branch in its search tree. That branch is infinite. The occurrence of infinite branches in some search trees creates troubles for implementations of Prolog based on a depth-first search strategy. An example is given in § 5.14.

5.8 EXAMPLE. Show that the search tree for the problem

$$ P(x) \leftarrow P(y) $$

$$ P(x) $$

has only one branch and that it is infinite.

5.9 Questions and Answers. It is possible to ask questions to and get answers from a Prolog program. We illustrate this by the example on the British Royal Family in §§ 4.26-28.

(I) If we want to know whether a variable-free atomic sentence $R$ follows from the program sentences, we ask by taking as input the question

$$ R $$

If Prolog succeeds in deducing $R$, the system answer 'Yes'. If Prolog does not succeed in deducing $R$, then the system answers 'No'.

**Question:**

$P(x, w)$

**Answer:**

Yes

**Question:**

$P(y, w)$

**Answer:**

No

**Is Elizabeth a paternal grandparent of William?**

**Is Diana a paternal grandparent of William?**
If we want to know whether there are individuals that satisfy the atomic formula $R(x,y,...)$, we take as input the question

$$
\therefore R(x,y,...)
$$

If there is a substitution which result in a deduction of $\therefore$, Prolog gives this substitution

$$
x = a, y = b, \ldots
$$

as the answer. If no substitution results in a deduction of $\therefore$, Prolog answers 'No'.

**Question:**

$\therefore P(x,w)$

**Answer:**

Philip is a paternal grandparent of William.

If we ask once more, Prolog continues the search in the search space until it finds another substitution which gives a deduction of $\therefore$.

**Question:**

$\therefore P(x,w)$

**Answer:**

Elizabeth is a paternal grandparent of William.

If we ask the same question a third time, Prolog does not succeed in finding a third substitution.

**Question:**

$\therefore P(x,w)$

**Answer:**

No

The answer 'Philip is a paternal grandparent of William' is given before the answer 'Elizabeth is a paternal grandparent of William' because we assume that Prolog applies a depth-first search strategy and goes through the search space from the left towards the right. Already the first branch gives the answer 'x = p'. Not until the fourth branch will the answer 'x = e' ensue. With another ordering of the program sentences, the answer 'Elizabeth is a paternal grandparent of William' will appear first.

5.10 Applications. The most important application of Prolog at present is presumably expert systems. An expert system answers questions or gives advice within a specialised field, e.g., medicine, finances, gastronomy, or library work.

A very simple example which shows how expert systems work in principle is given in § 4.26-27. The sentences (P1)-(P4) contain basic facts and constitute the data base. (P5)-(P6) give a definition which makes it possible to get out information stored deeper in the data base.

5.11 REMARK. Prolog is without any doubt a considerable step forward in computing science and artificial intelligence. In spite of this, Prolog still have a number of deficiencies. We mention three here.

5.12 Horn Clauses. All algorithms can be formulated in predicate logic and therefore also in clauses. All algorithmic problems can also be formulated and solved in Horn sentences in spite of the fact that Horn sentences only are a fragment of predicate logic. This follows since all recursive functions can be expressed and calculated in Horn sentences. All algorithms can be represented by recursive functions. This connection gives a method of formulating all algorithms, and thus all programs, in Prolog.

Apparently all problems can be formulated in predicate logic, and therefore also in clause logic. Every problem in predicate logic or general clause logic and the search in any of these systems for a solution to the problem can be formulated as an algorithmic problem. Since all algorithms can be formulated and executed in Horn clause logic, all problems can be formulated in Prolog, and every problem which has a solution can, in principle, be solved in Prolog. Programs formulated in Horn clauses by these indirect methods are, however, extremely slow compared to programs formulated by the direct methods which we studied earlier in this chapter. In practice, there are problems which are outside the scope of Prolog. This implies a practical limitation of the applicability of Prolog.

5.13 Unsoundness. The occur check is part of the complete unification algorithm. In § 4.23, we saw how the occur check makes it necessary all the time to create new variants of the clauses to be matched and possibly resolved. This contributes to making the unification algorithm, and thus Prolog, slow.

To speed up the unification algorithm, the occur check is omitted in many implementations of Prolog. This gives, however, rise to unsoundness in the system as the following example shows.
We now try to unify the \( y \) and \( f(\text{y}) \) in the second place in (2) and (3). With the occur check present, it should be impossible. Since it is omitted, the unification algorithm produces the following substitution
\[
\{ y \leftarrow f(y) \}
\]
(1) \( P(x, f(x)) \leftarrow P \)
(2) \( \leftarrow P(y, y) \)
(3) \( P(y, f(y)) \leftarrow 1. \text{Unification of } x \text{ and } y \)

(4) \( \leftarrow P(f(y), f(y)) \)
(5) \( P(f(y), f(f(y))) \leftarrow 2. \text{"Unification" of } y \text{ and } f(y) \)
(6) \( \leftarrow 3. \text{"Unification" of } y \text{ and } f(y) \)

Since \( f(y) \neq f(f(y)) \), (4) and (5) can, strictly speaking, not be resolved with each other. But since the unification algorithm did not answer 'False' when it attempted a unification of \( P(y, y) \) and \( P(y, f(y)) \), Prolog considers \( P(f(y), f(f(y))) \) and \( P(f(y), f(f(y))) \) as unified. Therefore (4) and (5) are resolved.

(6) \( \leftarrow 4, 5, \text{"Resolution"} \)

This is wrong. The premises (1) and (2) are consistent with each other as is brought out by the model
\[
\mathcal{M} = (\mathbb{N}, P, f) = (\omega, <, S)
\]
where \( \mathbb{N} = \{0, 1, 2, \ldots \} \) and \( S(x) = x+1 \) is the successor operation.

It must be pointed out that logic programming in itself is sound. The unsoundness just pointed out belongs to certain implementations of Prolog. It should also be pointed out that there are implementations of Prolog where the occur check is not omitted from the unification algorithm. Such implementations do not suffer from the named unsoundness.

5.14 Incompleteness. Clause logic is complete. The same is true of clause logic limited to Horn clauses. If a clause is a logical consequence of a set of premises, then there is also a deduction in clause logic which demonstrates this. Prolog, as based on a depth-first search strategy, is, however, not complete in this sense as is brought out by the following example.

**Program sentence:**

(R1) \( P(a, b) \leftarrow \)
(R2) \( P(b, c) \leftarrow \)
(R3) \( P(c, a) \leftarrow P(x, y), P(y, x) \)

(R4) \( P(x, y) \leftarrow P(y, x) \)

Goal:
\[
\leftarrow P(a, b)
\]

It is easy to deduce \( \leftarrow \) from these premises. There is a branch in the search tree for the problem which ends in \( \leftarrow \). Unfortunately, Prolog will never find this branch. Applying a depth-first search strategy, Prolog gets lost in an infinite branch.

Generally, a logic programming language with a search strategy based on depth-first is incomplete. Implementations of Prolog with a breadth-first search strategy are complete. What is problematic about a breadth-first algorithm is that it is too slow. Logic programming therefore has to choose between inefficiency and incompleteness. A logic programming language designed for practical purposes will therefore unavoidably be incomplete.

Some implementations of Prolog based on the depth-first strategy stops examining any branch longer than \( N \) steps, for some number \( N \), when they reach the \( N \)th step. But even such implementations of Prolog are incomplete. No matter how large \( N \) is, there are always solvable deduction problems where the shortest valid deduction is longer than \( N \) steps.

5.15 EPILOG. Even with these shortcomings, Prolog is a considerable resource in the AI research and technology. In several respects, Prolog is more natural and convenient than other programming languages. Some of the problems mentioned above and connected with the slowness of Prolog can possibly be solved by faster and more powerful computers. The future of Prolog depends on whether it is possible to find other and better ways to implement logic in artificial systems. The ability to make logical inferences is an essential component of all intelligence that it is impossible to imagine artificial intelligence without logic. Logic will therefore be a part of AI research and technology even in the future, either in the form of Prolog or possibly in some other form still waiting for discovery.

The form of logic which AI is waiting for may be the informal logic we envisaged in Chapter 2. Human beings do not normally use formal logic when they make inferences. Because that would imply, as shown in Chapter 2, that implications are based on functional connec-
tions at the object language level; and normally our implications are based on functional connections at the object level. The human mind is enormously superior to all artificial systems in the conceptual processing of information. This is just another way of saying that human beings are much more intelligent than any known artificial system. A first step towards artificial systems with real intelligence may be to endow the systems with the kind of logic we use rather than the mechanical formal logic favoured now. Formal logic, in the form of clause logic, has had a quick and easy success in AI. Like so many other quick and easy successes, this may turn out to be a cul-de-sac.

9-6 Completeness Theorems

6.1 DEFINITION. A Horn clause logic (HCL) is a logic determined by the following:
(1) a predicate logical language L;
(2) the set of all Horn clauses in L;
(3) the class of all identity axioms of L;
(4) resolution and unification as deduction rules.

6.2 DEFINITION. A general clause logic (GCL) is a logic determined by the following:
(1) a predicate logical language L;
(2) the set of all clauses in L;
(3) the class of all identity axioms of L;
(4) resolution, contraction, and unification as deduction rules.

6.3 DEFINITION. By Prolog we mean an automated form of Horn clause logic determined by the following:
(1) a predicate logical language L;
(2) the set of all Horn clauses in L;
(3) the class of all identity axioms for L;
(4) backward resolution and unification as deduction rules;

(5) an algorithm for the backward resolution operation; it is understood that this algorithm is based on a breadth-first method for search in the search space;
(6) the unification algorithm in Section 4.

6.4 PROBLEM. Since HCL and GCL are fragments of predicate logic, the soundness of HCL and GCL follows from the soundness of predicate logic. In this section, we want to prove a number of completeness results:
(1) Prolog and HCL are of equal power.
(2) HCL is complete relative to the usual set theoretic semantics.
(3) GCL is complete relative to the usual set theoretic semantics.
(4) Every recursive function can be computed in HCL.
(5) Prolog is complete as a functional programming language.
(6) Every problem, which can be formulated and solved in predicate logic, can be formulated and solved in GCL.
(7) Every problem, which can be formulated and solved in predicate logic, can be formulated and solved by HCL.
(8) Every problem, which can be formulated and solved in predicate logic, can be formulated and solved in Prolog.

6.5 NOTATION. Let S be a set of clauses in a clause logic CL. Then we use
\[ S \vdash P_1, ..., P_m \iff Q_1, ..., Q_n \]
to denote that there is a derivation of \( P_1, ..., P_m \iff Q_1, ..., Q_n \) from S in the logic CL (which may be GCL, HCL, or Prolog).

6.6 LEMMA. Let S be a set of Horn clauses. Then S \( \vdash \iff \) can be proved in HCL iff S \( \vdash \iff \) can be proved in GCL.

PROOF:
\[ s \vdash \iff \]
From the formulation of the resolution rule we see that every clause in a deduction in GCL must be a Horn clause. Therefore the contraction rule is never used.

\[ s \vdash \iff \]
Trivial.
6.7 LEMMA. Assume that there is a deduction of \( S \vdash \leftarrow \) in HCL. Then there is a deduction of \( S \vdash \leftarrow \) in HCL where backward resolution is the only form of resolution used.

PROOF:
Suppose that there is no deduction of \( S \vdash \leftarrow \) in HCL using only backward resolution. Among the deductions of \( S \vdash \leftarrow \), take one with fewest possible applications of non-backward resolution. Let this number be \( (s+1) \). In this deduction, consider the last application of a case of non-backward resolution:

\[
\begin{align*}
(1) & \quad T \leftarrow Q_1, \ldots, Q_n \\
(1+1) & \quad R \leftarrow S_1, \ldots, S_m, T \\
(1+2) & \quad R \leftarrow Q_1, \ldots, Q_n, S_1, \ldots, S_m
\end{align*}
\]

Remove line \((1+2)\) from the deduction. Then we at the same time decrease the number of non-backward resolutions in the deduction by one.
If line \((1+2)\) is not used later in the deduction, the modified sequence is still a deduction which terminates in \( \leftarrow \). This contradicts the minimality of \((s+1)\). If line \((1+2)\) is used later, consider any such application:

\[
\begin{align*}
(2) & \quad R \leftarrow Q_1, \ldots, Q_n, S_1, \ldots, S_m \\
(2+1) & \quad U_1, \ldots, U_m \\
(2+2) & \quad Q_1, \ldots, Q_n, S_1, \ldots, S_m, U_1, \ldots, U_m
\end{align*}
\]

This must be a case of backward resolution since the one on line \((1)+(2+)\) is the last non-backward resolution in the deduction. Replace lines \((2)-(2+)\) by:

\[
\begin{align*}
(2') & \quad R \leftarrow S_1, \ldots, S_m, T \\
(2'+1) & \quad U_1, \ldots, U_m \\
(2'+2) & \quad S_1, \ldots, S_m, U_1, \ldots, U_m, T
\end{align*}
\]

We see that the case of backward resolution using \((1+2)\) in lines \((2)-(2+)\) has been replaced by two backward resolution operations in \((2')-(2'+3)\) which do not use \((1+2)\). Thus we have transformed the original deduction of \( \leftarrow \) with \((s+1)\) applications of non-backward resolution into a deduction of \( \leftarrow \) with only \( s \) applications of non-backward resolution. This contradicts the minimality of \((s+1)\).

6.8 THEOREM. HCL and Prolog are equivalent systems, i.e., for every finite set \( S \) of Horn clauses,
\( S \vdash \leftarrow \) in Prolog \( \iff \) \( S \vdash \leftarrow \) in HCL.

PROOF:
Every deduction in Prolog is also a deduction in HCL.
Lemma 6.7 shows that the limitation in Prolog to backward resolution does not make Prolog a weaker logic than HCL. It is also clear that there is an algorithm for backward resolution which by systematically searching the search space breadth-first finds a proof whenever there is one. The only remaining problem is to show that the unification algorithm is logically as powerful as the general unification rule.

Checking the unification algorithm, it is clear that the algorithm finds a most general unifier (MGU). Consider a deduction of \( S \vdash \leftarrow \) in HCL using the general unification rule. By Lemma 6.7, we may assume that the deduction uses only backward resolution. We show that there is a deduction of \( S \vdash \leftarrow \) in Prolog of the same length such that every clause in the HCL deduction is an instance of a variant of the corresponding clause in the Prolog deduction. Assume that this is satisfied for every line in the deduction before line \( k \). Consider the next step in the HCL deduction:

\[
\begin{align*}
(1) & \quad T \leftarrow P_1, \ldots, P_m \\
(1) & \quad \\n(2) & \quad \leftarrow Q_1, \ldots, Q_n, S \\
(3) & \quad T \leftarrow P_1, \ldots, P_m, S \\
(3) & \quad \leftarrow Q_1, \ldots, Q_n, S \\
(3) & \quad \leftarrow Q_1, \ldots, Q_n, S
\end{align*}
\]

We see that line \( k \) in the Prolog deduction is an instance of the corresponding line \( k \) in the HCL deduction. By induction hypothesis, lines \( i, j \) in the HCL deduction are instances of variants of lines \( i, j \) in the Prolog deduction. By changing bound variables, we may assume that they are instances of the corresponding lines in the Prolog deduction. Then \( T \) is an instance of \( K \) and \( S \) is an instance of \( R \). It follows that \( T^* \) is an instance of \( K \) and \( S^* \) is an instance of \( R \). Since \( K \) and \( R \) can be unified into \( T^* = S^* \), \( K \) and \( R \) can be unified by a
most general unifier. Call the result of this most general unification of \( K \) and \( R \) for \( K^{**} = R^{**} \). Thus the step to line (k+2) in the Prolog deduction can be performed. Since the \( ** \)-substitution is at least as general as the \( * \)-substitution, lines \( k \), \( (k+1) \), \( (k+2) \) in the HCL deduction are instances of variants of the corresponding lines in the Prolog deduction.

We see that there is a valid deduction of \( S \vdash \text{in} \) which can be done by using only MGU. When Prolog systematically searches the search space, it will find such a deduction of \( S \vdash \text{in} \).

**6.9 Theorem.** GCL is complete relative to the semantics of predicate logic, i.e., for every set \( S \) of clauses,

\[ S \vdash \text{in} \iff S \vdash \text{in} \text{GCL} \]

**Proof:**

- **step:** This follows immediately since GCL is a fragment of predicate logic.

- **step:** Since ordinary predicate logic is complete, it suffices to show that GCL has the same deductive power as predicate logic restricted to (the standard form of) clauses. We compare with a system of natural deduction. We first consider the specific predicate logical rules (VI), (VE), (EI), (EI), and the rules for identity, (Ref) and (Suba). Since all identity axioms are in GCL, the operations for identity can be performed in GCL. The rules for existential quantification (II) and (EI) are not relevant since the existential quantifiers in clauses are replaced by Skolem functions. The (VI)-operation is in GCL done automatically since it is understood that all variables in clauses are bound by universal quantifiers. This implies that the full strength of the (VI)-operation is present in GCL. The use of unification is a case of (VI); but it does not imply the full scope of (VE). The substitutions in the (VE)-rule applied in GCL are limited by the demand that they must lead to a matching of two atomic formulas. Note that unification in itself does not lead to an elimination of an occurrence of an atomic formula in a deduction in GCL. This can only be performed by the rules resolution and contraction in GCL. The necessary condition for an application of any of them is that the eliminated occurrences match. Therefore an addition of (VE) to GCL should not give a system which is stronger than GCL, i.e., there is no set \( S \) of clauses such that

\[ S \vdash \text{in} \text{GCL}, \text{but } S \vdash \text{in} \text{GCL + (VE)} \]

We can therefore conclude that if sentential GCL is complete, then also the full predicate GCL is complete. We now consider the completeness of sentential GCL and the two sentential operations of resolution and contraction.

Let \( S \) be any set of classes in a language \( L \) consisting of a countably infinite sequence \( P_0, P_1, P_2, \ldots \) of sentential parameters. We now prove for sentential GCL.

\[ (6-1) \quad S \vdash \text{in} \iff S \vdash \text{in} \]

Assume the hypothesis in (6-1). We define sets \( M_k \) and \( M \) of classes:

\[ M_0 = S \]

\[ M_{k+1} = \begin{cases} M_k \cup \{ P_k \text{ in} \} & \text{if } M_k \cup \{ P_k \text{ in} \} \vdash \text{in} \\ M_k \cup \{ \text{in} \} & \text{otherwise} \end{cases} \]

\[ M = \cup_k M_k \]

We prove by induction on \( k \).

\[ (6-2) \quad M_k \vdash \text{in} \iff \text{for all } k \]

This is by the hypothesis of (6-1) true for \( M_0 = S \). Assume that (6-2) has been proved for \( M_k \), but suppose that \( M_{k+1} \vdash \text{in} \). By the definition of \( M_{k+1} \):

\[ (6-3) \quad M_k, P_k \vdash \text{in} \]

\[ (6-4) \quad M_k, \text{in} \vdash P_k \]

Consider any deduction of (6-4). A resolution operation where \( \text{in} \vdash P_k \) is a premise

\[ (1) \quad \text{in} \vdash P_k \]

\[ (2) \quad Q_1, \ldots, Q_m, P_k \vdash R_1, \ldots, R_n \]

\[ (3) \quad Q_1, \ldots, Q_m \vdash R_1, \ldots, R_n \quad 1, 2, \text{resolution} \]

has the sole effect of eliminating one occurrence of \( P_k \) in the consequent of the other premise. Remove all such applications of \( \text{in} \vdash P_k \) from the deduction. This gives a deduction of

\[ (6-5) \quad M_k \vdash P_k, \ldots, P_k \]

By the contraction rule,

\[ (6-6) \quad P_k \]

228

229
But then by (6-3),

\[(6-7) \quad M_k \vdash \leftarrow\]

which contradicts the induction hypothesis. This concludes the proof of the induction step so that (6-2) is true. From (6-2) follows immediately

\[(6-8) \quad M \vdash \leftarrow\]

From (6-8) and the definition of the $M_k$, we have for any $k = 0, 1, 2, \ldots$

\[(6-9) \quad \forall k \vdash M \text{ or } \forall k \vdash M \text{ but not both.}\]

The result (6-9) implies that $M$ is a model for GCL with the truth condition

\[(6-10) \quad M \models Q_1, \ldots, Q_n \leftarrow R_1, \ldots, R_m \iff Q_i \leftarrow \text{ M for some } i \text{ or } R_j \leftarrow \text{ M for some } j\]

or equivalently

\[(6-11) \quad M \models Q_1, \ldots, Q_n \leftarrow R_1, \ldots, R_m \iff \leftarrow Q_1', \ldots, \leftarrow Q_n', R_1' \leftarrow \ldots, R_m' \leftarrow \text{ M}\]

We prove that $M$ is a model of $S$, i.e.,

\[(6-12) \quad M \models S\]

Suppose (6-12) is false. Then for some clause in $S$,

\[M \not\models Q_1, \ldots, Q_n \leftarrow R_1, \ldots, R_m\]

By (6-11),

\[(6-13) \quad \leftarrow Q_1', \ldots, \leftarrow Q_n', R_1' \leftarrow \ldots, R_m' \leftarrow \text{ are in M}\]

Since $S \subseteq M$,

\[(6-14) \quad Q_1', \ldots, Q_n' \leftarrow R_1', \ldots, R_m' \text{ is in M}\]

By iterating the resolution operation on the clauses in (6-13) and (6-14), we get

\[(6-15) \quad M \vdash \leftarrow\]

which contradicts (6-8). Therefore (6-12) must be true. Since $S$ has a model, $S$ is consistent. Therefore $S \models \bot$, and thus $S \models \leftarrow$. This concludes the proof of (6-1).

We can now complete the proof of the theorem. Let $S$ be a set of clauses in a predicate logical language $L$. Assume $S \models \leftarrow$.

Let $GCL^*$ be $GCL$ extended by the (VE)-rule. We use $\vdash^*$ to denote the deduction relation in $GCL^*$. Let $S$ be the set of all substitution instances in $L$ of classes in $S$. Then $S^* \models \leftarrow$.

By the completeness of natural deduction for the standard form of predicate logic $S^* \vdash \leftarrow$.

The only predicate logical rule which can be applied in a deduction of $\leftarrow$ from $S^*$ is (VE). But since $S^*$ is closed under substitution instances, (VE) is not needed. Therefore in turn,

\[S^* \vdash \leftarrow\]

by natural deduction

By sentential logic in the standard form $S^* \models \leftarrow$.

By (6-1), we can infer $S^* \models \leftarrow$.

By sentential GCL.

Clearly,

\[S \vdash^* S^*\]

Combining (6-16) and (6-17), we have

\[S \vdash^* \leftarrow\]

As shown above, $GCL^*$ and $GCL$ are of equal strength so that

\[S \models \leftarrow \text{ in } GCL\]

Thus

\[S \models \leftarrow \Rightarrow S \models \leftarrow \text{ in } GCL\]

6.10 THEOREM. HCL is complete relative to the semantics of predicate logic, i.e., for every set $S$ of Horn clauses, $S \models \leftarrow \iff S \models \leftarrow$ in HCL.

PROOF:

Use Theorem 6.9 and Lemma 6.6.
6.11 THEOREM. Prolog is complete relative to the semantics of predicate logic, i.e., for every finite set $S$ of Horn clauses,

$$S \vdash \Rightarrow$$

$S \\vdash \Rightarrow$ in Prolog.

PROOF:

Use theorems 6.8 and 6.10.

6.12 REMARK. The next goal is to prove that Prolog is complete as a functional programming language, i.e., it is computationally at least as powerful as such programming languages like Pascal, C, and ML. This will follow as soon as we have proved that every algorithm can be represented and executed in Prolog. The concept of an algorithm is intuitive and nonmathematical. As a first step, we prove that every recursive function can be computed in HCL and Prolog. I will take pains to show by an analysis that every algorithm can be represented by a recursive function. Normally this is done by using Turing machines and, e.g., in Lewis and Papadimitriou (1981). Here I will proceed in a somewhat different way. In the analysis, the following concepts are used:

1. algorithm;
2. algorithm over alphabet;
3. normal algorithm over alphabet;
4. computable function;
5. recursive function.

These concepts will now be defined.

6.13 Algorithms. Following Knuth (1973), an algorithm may be characterised as a finite set of rules which, given any problem in a specific class of problems, yields a sequence of operations which solves the problem and which satisfies the following conditions:

1. Finiteness. An algorithm must halt after a finite number of steps.
2. Definiteness. Each step of an algorithm must be precisely defined.
3. Input. An algorithm must have zero or more inputs. The set of inputs of an algorithm consists of the entities which can be fed into the algorithm. The input is the entity on which the algorithm makes the first operation in its sequence of operations.
4. Output. An algorithm has one or more outputs. An output is an entity which is the result of the last operation in the sequence of operations for

the given problem. The output is the solution given by the algorithm to the problem under treatment.

5. Effectiveness. An algorithm must be effective, i.e., every operation in its sequence of operations must be concrete so that it can in principle be done exactly and in a finite length of time by a human being.

6.14 DEFINITION. An algorithm is a quadruple

$$A = (Q, I, \Omega, \varphi)$$

consisting of

- $Q$: the states of computation;
- $I$: the set of inputs;
- $\Omega$: the set of outputs;
- $\varphi$: the computational role.

They satisfy

$$I, \Omega \subseteq Q$$

$$\varphi : Q \rightarrow Q$$

Each input $x$ determines a unique finite sequence

$$x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_n$$

where

$$x_0 = x$$

is the input;

$$\varphi(x_k) = x_{k+1}$$

for $0 \leq k \leq n$;

$x_n$ is the terminal object or output.

Thus $A$ is a mapping

$$A : I \rightarrow \Omega$$

the operation of which on each sequence can be analysed, in terms of the effect of $\varphi$, into a sequence of discrete steps.

6.15 REMARK. (I) The definition is intuitive and nonmathematical. The reason for this is that there is no exact definition of which functions $\varphi$ qualify as computational rules.

(II) An algorithm may be numerical or nonnumerical. It may be concerned with mathematical or nonmathematical entities.
6.16 DEFINITION. Let \( f: \mathbb{N}^n \rightarrow \mathbb{N} \) be a number theoretic function. \( f \) is a computable function \( \iff \) there is an algorithm \( A = (Q, I, \Omega, \delta) \) with \( I = \mathbb{N}^n \) and \( \Omega = \mathbb{N} \) such that for every \( a = (a_1, \ldots, a_n) \in \mathbb{N}^n \), \( A(a) = f(a) \).

6.17 DEFINITION. The recursive functions are defined by the following inductive definition.

(RF1) Basic functions: The zero function
\[ Z: \mathbb{N} \rightarrow \mathbb{N} \]
\[ Z(x) = 0 \]
the successor function
\[ S: \mathbb{N} \rightarrow \mathbb{N} \]
\[ S(x) = x + 1 \]
the projection functions
\[ U_{n,y}: \mathbb{N}^n \rightarrow \mathbb{N} \]
\[ U_{n,y}(x_1, \ldots, x_n, y) = x_y \]
are recursive.

(RF2) Substitution: Let
\[ h_i: \mathbb{N}^n_i \rightarrow \mathbb{N} \text{ for } 1 \leq i \leq n \]
\[ g: \mathbb{N}^n \rightarrow \mathbb{N} \]
be recursive. Then
\[ f: \mathbb{N}^n \rightarrow \mathbb{N} \]
\[ f(x) = g(h_1(x), \ldots, h_n(x)) \]
is recursive where \( x = (x_1, \ldots, x_m) \).

(RF3) Primitive recursion: Let
\[ h_2: \mathbb{N}^{n+1} \rightarrow \mathbb{N} \]
\[ g: \mathbb{N}^n \rightarrow \mathbb{N} \]
be recursive. Then
\[ f: \mathbb{N}^{n+1} \rightarrow \mathbb{N} \]
\[ f(x, 0) = g(x) \]
\[ f(x, y + 1) = h(x, y, f(x, y)) \]
is recursive.

(RF4) Minimalisation: Let \( h: \mathbb{N}^{n+1} \rightarrow \mathbb{N} \) be recursive, and assume that
\[ \forall x_1 \ldots \forall x_n \exists y h(x_1, \ldots, x_n, y) = 0 \]
Then
\[ f: \mathbb{N}^n \rightarrow \mathbb{N} \]
\[ f(x) = \min \{ y : h(x_1, \ldots, x_n, y) = 0 \} \]
is recursive where \( \min \{ \ldots \} \) denotes the smallest number \( y \in \mathbb{N} \) which satisfies the condition \( \ldots = 0 \).

6.18 THESIS (Church-Turing). Let \( f: \mathbb{N}^n \rightarrow \mathbb{N} \) be a function. Then \( f \) is computable \( \iff \) \( f \) is recursive.

6.19 ANALYSIS. (I) The direction \( \Leftarrow \) can be proved mathematically. The basic functions in (RF1) are trivially computable. We also note that the rules (RF2)-(RF4) only generate computable functions given that we start with computable functions.

(II) No proof for the direction \( \Rightarrow \) is known. It may be that no proof is possible due to the informal character of the concept of a computable function. Thesis 6.8 is usually treated as an empirical hypothesis. The evidence for the thesis is the following.

(1) Many attempts have been made by different mathematicians from different starting points to define mathematically what a computable function is. Examples are recursive functions (Herbrand-Gödel-Kleene), \( \lambda \)-definable functions (Church), Turing machines (Turing), normal algorithms (Markov), and functions which are representable in arithmetic (Gödel). These definitions have been proved mathematically to be equivalent.

(2) We may try to analyse what a computation by hand, using pen and paper, consists in. Such an analysis was made by Turing (1936). A modern exposition can be found in Lewis and Papadimitriou (1981).
It leads to the theory of Turing machines. It can be proved that a function is recursive iff it is Turing computable.

(3) No computable function has ever been discovered in the history of mathematics which is not also recursive.

6.20 DEFINITION. (I) An alphabet $A$ is a finite set of symbols called letters. A word in $A$ is a finite sequence

$$x_1 \ldots x_k$$

where $x_i \in A$ for $1 \leq i \leq k$

The set of all words in $A$ is denoted by $A^*$. (II) An algorithm in the alphabet $A$ is an algorithm $\mathcal{A} = (Q, I, \Omega, s)$ where $Q = \{1 = \delta^*\}$ and $\Omega \subset A^*$. A normal algorithm in the alphabet $A$ is meant to be an exact mathematical counterpart of an algorithm in $A$. Those interested in a detailed definition may consult Mačcev (1970).

6.21 THESIS (Normalisation principle). Let $\mathcal{A}$ be any algorithm in the alphabet $A$. Then there is a normal algorithm $\mathcal{B}$ over $A$ such that for any word $w \in A^*$,

$$\mathcal{A}(\mathcal{B}) = (\mathcal{B})$$

("There is considerable evidence for this analogue of Church-Turing's thesis. Some of this evidence is summarised in Analysis 6.24.*)

6.22 DEFINITION. (I) Let $\mathcal{A} = (Q_1, I_1, \Omega, s_1)$ and $\mathcal{B} = (Q_2, I_2, \Omega, s_2)$ be algorithms. Then $\mathcal{A}$ represents $\mathcal{B}$ if there is a constructive and injective function (an algorithm) $\Phi: Q_1 \rightarrow Q_2$ such that for all $x \in I_1$,

$$\mathcal{A}(x) = \Phi^{-1}(\mathcal{B}(\Phi(x)))$$

(II) In particular, the function $f: \mathbb{N} \rightarrow \mathbb{N}$ represents the algorithm $\mathcal{A} = (Q, I, \Omega, s)$ if there is constructive injection $\Phi$ such that for all $x \in I$,

$$\mathcal{A}(x) = \Phi^{-1}(f(\Phi(x)))$$

6.23 THESIS. Every algorithm can be represented by a recursively defined function.

6.24 ANALYSIS. There is no proof of the thesis but some empirical evidence.

6.25 DEFINITION. $f: \mathbb{N}^0 \rightarrow \mathbb{N}$ is primitive recursive $\iff f$ can be defined using only (RF1)-(RF3) in Definition 6.17.

6.26 DEFINITION. Let $R \subset \mathbb{N}^0$ be a relation. The characteristic function $K_R: \mathbb{N}^0 \rightarrow \mathbb{N}$ of $R$ is defined by

$$K_R(x) = \begin{cases} 0 & \text{if } R(x) \\ 1 & \text{if } \neg R(x) \end{cases}$$

6.27 DEFINITION. (I) The relation $R \subset \mathbb{N}^0$ is recursive $\iff$ its characteristic function $K_R$ is recursive. (II) $R$ is primitive recursive $\iff K_R$ is primitive recursive.

6.28 DEFINITION. We define the following functions in $\mathbb{N}$.

$$\text{Sign}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

$$x + y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

236

237
6.29 LEMMA. The following functions and relations are all primitive recursive:

\[ \text{Sign, } +, \cdot, \cdot, K_{<}, K_{=}, K_{<} \cdot, <, =, * \]

PROOF:
(1) Sign can be defined by primitive recursion:

\[ \text{Sign}(0) = 0 \]
\[ \text{Sign}(x+1) = 1_{2,2}(S(Z(x)), \text{Sign}(x)) \]

(2) To define \( x + y \), we first consider the function \( x + 1 \):

\[ 0 + 1 = 0 \]
\[ (x+1) + 1 = x \]

Thus \( x + 1 \) is primitive recursive. Now it is easy to give a primitive recursive definition of \( x + y \):

\[ x + 0 = x \]
\[ x + (y+1) = (x + y) + 1 \]

(3) \( x + y \) is defined by (RF3):

\[ x + 0 = x \]
\[ x + (y+1) = S(x + y) \]

(4) \( x \cdot y \) is defined by

\[ x \cdot 0 = 0 \]
\[ x \cdot (y+1) = x \cdot y + x \]

(5) Finally, we define \( K_{<}, K_{=}, \text{ and } K_{<} \cdot \):

\[ K_{<}(x,y) = S(0) + \text{Sign}(y+x) \]
\[ K_{=}(x,y) = \text{Sign}(x+y) + (y=x) \]
\[ K_{<}(x,y) = S(0) + K_{=}(x,y) \]

It follows that \( <, =, \) and \( \cdot \) are primitive recursive relations.

6.30 THEOREM (Tůrnai-Sebeřánek-Stepánková). Every recursive function can be computed in HCL.

PROOF:
The proof is by induction on the definition of recursive functions. We operate in the language \( L(PA) = \{ 0, S, +, \cdot, < \} \) of Peano arithmetic.

(RF1): Define the zero-function \( Z \) by

\[ Z(x) = 0 \leftarrow \]

Since in \( L(PA) \) the number \( n \) is represented by \( SS...SS \) where \( S \) occurs \( n \) times in the sequence, the equation

\[ S(x) = x + 1 \]

is an instance of the identity axiom

\[ x = x \leftarrow \]

If we represent \( S \) by the 2-place predicate \( S \), then \( S \) is defined by the possibly more satisfactory clause

\[ S(x,S(x)) \leftarrow \]

The projection function \( U_{\langle x \rangle} \) is defined by

\[ U_{\langle x \rangle}(x_1,...,x_n) = x_1 \leftarrow \]

(RF2): Let \( f \) be defined by composition from \( g, b_1, ..., b_n \). By the induction hypothesis, there are Horn clauses which define \( g, b_1, ..., b_n \). Define

\[ f(x) = y \leftarrow \quad b_1(x) = y_1, ..., b_n(x) = y_n, g(y_1,...,y_n) = y \]

(RF3): By the induction hypothesis, there are Horn clause procedures for computing \( h \) and \( g \). Then \( f \) is defined by the Horn clause procedure

\[ f(x,0) = z \leftarrow z = g(x) \]
\[ f(x,y+1) = z \leftarrow z = h(x,y,f(x,y)) \]

We have now shown that all primitive recursive functions are HCL-computable. All primitive recursive functions can therefore be used in the computations in HCL of recursive functions defined by minimalisation. In particular, the functions in Lemma 6.29 can be so used.

(RF4): Let \( f \) be defined by

\[ f(x) = \mu y (h(x,y) = 0) \]

where \( h \) is recursive and satisfies \( \forall x \exists y h(x,y) = 0 \). By the induction hypothesis, \( h \) can be computed in HCL. We define a function \( P_h(x,y) \) by

\[ P_h(x,0) = z \leftarrow z = \text{Sign}(h(x,0)) \]
\[ P_h(x,y+1) = z \leftarrow z = P_h(x,y) \cdot \text{Sign}(h(x,S(y))) \]
family. In practice, however, Prolog is often slower than functional languages in the execution of algorithms.

6.35 DEFINITION. Let \( L \) be a language and \( S \) a set of sentences in \( L \). Let \( T(S) \) be the theory in \( L \) whose set of nonlogical axioms is \( S \). A problem in \( T(S) \) is a question whether a given sentence \( A \) in \( L \) is true or false in \( T(S) \). If \( S \vdash A \) or \( S \vdash \neg A \), then the problem is solvable. If \( S \vdash A \) and \( S \vdash \neg A \), then the problem is unsolvable.

6.36 THEOREM. Let \( S \) be a set of sentences in \( L \). Every problem in \( T(S) \) which can be formulated and solved in predicate logic can be formulated and solved in GCL.

PROOF: Assume \( S \vdash B \) in predicate logic. Then \( S, \neg B \vdash \perp \). Transform \( S, \neg B \) into clause form as \( S^*, (\neg B)^* \). By Theorem 2.20,

\[ S^*, (\neg B)^* \vdash \perp \]

By Theorem 6.9,

\[ S^*, (\neg B)^* \vdash \perp \text{ in GCL} \]

Thus GCL gives the same answer to the given problem as predicate logic does, namely that \( A \) is true in \( T(S) \) if \( B = A \) and that \( A \) is false in \( T(S) \) if \( B = \neg A \).

6.37 THEOREM. Let \( S \) be a recursive set of sentences in a recursive language \( L \). Every problem in \( T(S) \) which can be formulated and solved in predicate logic can be formulated and solved in HCL.

PROOF: Since \( S \) is recursive, \( T(S) \) is an axiomatised theory. It follows that the proof predicate \( \text{Pr}_T(S) \) of \( T(S) \) is recursive. Then there is a recursive function \( F \) such that

\[ \text{Pr}_T(S) = \{ F(n) \mid n \in \mathbb{N} \} \]

with \( F(n) < F(n+1) \). By Theorem 6.30, \( F \) can be calculated in HCL.

Assume \( S \vdash B \) in predicate logic. We must show that the piece of information that \( S \vdash B \) can also be obtained in HCL. Take the proof of \( B \) from \( S \) with the smallest Gödel number, i.e., the first \( F(n) \) which is a proof of \( B \) in \( T(S) \). Since \( F \) is recursive, this Gödel number \( F(p) \) can be computed in HCL as follows. Calculate in turn \( F(0), F(1), \ldots \) until the
first Gödel number of a proof of B comes up. The only problem is to
decide in HCL for each of F(x) whether it is a proof of B or not, i.e.,
whether the last component in the sequence number F(n) is the Gödel
number b of B. This can be done by the following method. If x is a
sequence number, then ln(x) is the length of x and (x)_y is the yth
component of x. These two functions are recursive (see Shoenfield (1967)).
By Theorem 6.30, they are HCL-computable. Consider a set C contai-
ning the defining Horn clauses for the functions F(x), I(x) and (x)_y.
Then
\[(6-32) \quad C, \leftarrow \langle F(x) \rangle_{I(x)} = b \leftarrow \quad \text{in HCL} \]
while for all n < p,
\[(6-33) \quad C, \leftarrow \langle F(n) \rangle_{I(x)} = b \leftarrow \quad \text{in HCL} \]
This gives a method to show in HCL that S |- B.

6.38 THEOREM. Let S be a recursive set of sentences in a recursive
language L. Every problem in T[S] which can be solved by predicate
logic can be solved in Prolog.


6.39 REMARK. From the proof of Theorem 6.37, we see that the
deductions in OCL or HCL from a given recursive set S of premises
can be represented by the calculation of one recursive function F_S, a
function which enumerates Th_{T[S]} (x). This shows that every deduction
which can be done in a logic programming language also can be done in
a functional programming language. Combining this observation with
Remark 6.34, we conclude:

\[(6-34) \quad \text{The class of logic programming languages and the class of functional programming languages are equivalent in power.} \]

This is a purely theoretical result of little practical importance.
Logic programs are almost always slower than functional programs in
calculating algorithms. Functional programs are much slower than logic
programs in deduction. For the same reason, the indirect methods used
to prove theorems 6.37 and 6.38 result in logic programs which are
many times slower than logic programs formulated by the direct
methods studied in sections 1-5. In practice, there are problems which
are outside the scope of Prolog.

6.40 REMARK. Almost all theorems in the present section have been
formulated independently by the author. Since they all answer very
natural questions, it is unlikely that any of them is new. All proofs have
been developed from scratch. They may contain some novelties. I have
found Theorem 6.30 in Lloyd (1987). The first proof of this theorem
was given by S.Å. Tarján in 1977 using Turing's definition of com-
putable functions. A proof based on the Herbrand-Gödel-Kleene defi-
nition of recursive functions was given by Sebelek and Stepanek in 1982.
Lloyd follows Sebelek and Stepanek. For the rules (RF1)-(RF3), my
proof of Theorem 6.30 agrees with theirs except for inessential details.
My proof for the rule (RF4) is very different from theirs and seems to be
new. Though my proof is not shorter than the Sebelek-Stepanek proof, it
is conceptually much simpler.

9-7 Model Theory of Horn Sentences

7.1 REMARK. In this section, we operate in the standard form of predic-
ate logic. We derive a model theoretic criterion for the possibility of
formulating a theory or a problem by Horn sentences only. Since Horn
clauses are only another way of writing Horn sentences, this is also, by
Theorem 6.10, a criterion of the possibility of formulating a given
problem in Horn clause logic.

7.2 NOTATION. Let \( M = (M, \ldots) \) be a model for a language L. Then
L_M denotes L expanded with a unique name of each element in M. Let
P, f, c \in L be a predicate, a function symbol, and a constant, respec-
tively. Then \( P_m, f_m, c_m \) are the interpretations of P, f, c in M. If it is
clear which model M is meant, the subscripts may be omitted. If we
consider a family of models \( \{ M_i \}_{i \in I} \) indexed by \( i \in I \), then \( P_x, f_x, c_x \)
are the interpretations of P, f, c in \( M_i \). We use
\[ M_0, M_1, M_2, \ldots, M_p, \ldots \]
to denote models. Normally, we write the domains of \( \mathbb{A}, \mathbb{B}, \mathbb{M} \) as \( A, B, M \), respectively.

7.3 DEFINITION. \( A \) is a submodel of \( \mathbb{B} \) (\( \mathbb{B} \) is an extension of \( A \)) iff

1. \( A \subseteq B \);
2. For \( P, f, e \in L \) and \( a_1, \ldots, a_n \in A \),
   \[ (a_1, \ldots, a_n) \in P_A \iff (a_1, \ldots, a_n) \in P_B \]
   \[ f_A(a_1, \ldots, a_n) = f_B(a_1, \ldots, a_n) \]
   \[ e_A = e_B \]

7.4 LEMMA. Let \( M_1 \) and \( M_2 \) be models for \( L \) such that \( M_1 \subseteq M_2 \). Then \( M_1 \) is a submodel of \( M_2 \) iff for every variable-free formula \( A \) of \( L(M_1) \),

\[ M_1 \models A \iff M_2 \models A \]

7.5 DEFINITION. (I) \( T \) is a subtheory of \( T' \) (\( T' \) is an extension of \( T \)) iff \( L(T) \subseteq L(T') \) and every theorem of \( T \) is a theorem of \( T' \).
(II) \( T \) and \( T' \) are equivalent iff they have the same language and the same theorems.

7.6 LEMMA. Let \( T \) and \( T' \) be theories in the same language \( L \).
(I) \( T' \) is an extension of \( T \) iff every model of \( T' \) is a model of \( T \).
(II) \( T \) and \( T' \) are equivalent iff they have the same models.

7.7 DEFINITION. \( T \) is an open theory iff all nonlogical axioms of \( T \) are quantifier-free.

7.8 THEOREM. (Los-Tarski). Let \( T \) be a theory. \( T \) is equivalent to an open theory iff every submodel of a model of \( T \) is a model of \( T \).

7.9 DEFINITION. Let \( I \) be a nonempty index set. For each \( i \in I \), let \( M^i \) be a model for the language \( L \). The direct product \( M = \prod_{i \in I} M^i \) is the model for \( L \) which satisfies the following conditions for \( c, f, P \in L \):

\[ M = \prod_{i \in I} M^i \]
\[ (c^i)_1 = c^i \text{ for all } i \in I \]
\[ (f^i_1(a_1, \ldots, a_n)) = f^i_1((a^i_1, \ldots, a^i_n)) \text{ for all } i \in I \]
\[ P^i((a^i_1, \ldots, a^i_n)) \iff P^i((a^i_1, \ldots, a^i_n)) \forall i \in I \]

7.10 DEFINITION. Let \( A, B \) be models for \( L \) and let \( \Phi : A \rightarrow B \) be a mapping. \( \Phi \) is a homomorphism from \( A \) to \( B \) iff for all \( c, f, P \in L \) and all \( a_1, \ldots, a_n \in A \),

\[ \Phi(c_A) = c_B \]
\[ \Phi(f_A(a_1, \ldots, a_n)) = f_B(\Phi(a_1), \ldots, \Phi(a_n)) \]
\[ P_A((a_1, \ldots, a_n)) \iff P_B(\Phi(a_1), \ldots, \Phi(a_n)) \]

7.11 LEMMA. Let \( \Phi \) be a homomorphism from \( A \) to \( B \). Then for every term \( t(x_1, \ldots, x_n) \) with \( x_1, \ldots, x_n \) as only variables and for all \( a_1, \ldots, a_n \in A \),

\[ \Phi(t_A(a_1, \ldots, a_n)) = t_B(\Phi(a_1), \ldots, \Phi(a_n)) \]
and for every atomic formula \( C(x_1, \ldots, x_n) \) having only \( x_1, \ldots, x_n \) free and for all \( a_1, \ldots, a_n \in A \),

\[ A \models C(a_1, \ldots, a_n) \iff B \models C(\Phi(a_1), \ldots, \Phi(a_n)) \]

7.12 LEMMA. Define \( M^i_k \), \( \forall i \), and their direct product \( \mathcal{M} \) as in Definition 7.9. For each \( i \in I \), define the projection function \( p_I : M \rightarrow M^i \) by \( p_i(a) = (a)_i \). Then \( p_I \) is a surjective homomorphism from \( \mathcal{M} \) to \( M^i_k \).

PROOF:
It is clear that \( p_I \) is surjective. To prove that it is a homomorphism, use Definition 7.10. As an example, we consider the function symbol \( f \). Using Definition 7.9 and the definition of \( p_I \), we have for \( a_1, \ldots, a_n \in M \),

\[ p_I(f^i_1(a_1, \ldots, a_n)) = f^i_1((a^i_1, \ldots, a^i_n)) \]
\[ = f^i_1(p^i_1(a_1), \ldots, p^i_1(a_n)) \]

which agrees with the hypothesis in Definition 7.10.

7.13 LEMMA. Let \( A(a_1, \ldots, a_n) \) be an atomic formula in \( L \) having only \( x_1, \ldots, x_0 \) free and let \( a_1, \ldots, a_n \in M \). Then

\[ A(a_1, \ldots, a_n) \iff \mathcal{M} \models A(p_I(a_1), \ldots, p_I(a_n)) \text{ for all } i \in I \]
PROOF:
A is of the form $P(t_1(x_1,\ldots,x_m),\ldots,t_k(x_1,\ldots,x_m))$ where the $t_i(x_1,\ldots,x_m)$ are terms with no other variables than $x_1,\ldots,x_m$ and $P$ is a logical or nonlogical predicate. Using Definition 7.9 and Lemmas 7.11 and 7.12, we get

$$\mathcal{M} \models A(x_1,\ldots,x_n) \iff \mathcal{M} \models P(t_1(x_1,\ldots,x_n),\ldots,t_k(x_1,\ldots,x_n))$$

$$\iff \mathcal{M}_i \models P(t_1(x_1,\ldots,x_n),\ldots,t_k(x_1,\ldots,x_n))$$

where $t_1,\ldots,t_k$ are terms over $x_1,\ldots,x_n$.

7.14 DEFINITION. (i) $A$ is a Horn formula if $A$ is of one of the forms

$$P_1 \land \ldots \land P_n \rightarrow Q$$

$$P_1 \land \ldots \land P_n \rightarrow \bot$$

where $P_1,\ldots,P_n, Q$ are atomic formulas and $n \geq 0$.

(ii) $A$ is a closed Horn formula if $A$ is a variable-free Horn formula.

(iii) $A$ is a Horn sentence if $A$ is of the form

$$\forall x_1 \ldots \forall x_n \ B(x_1,\ldots,x_n)$$

where $B(x_1,\ldots,x_n)$ is a Horn formula having only $x_1,\ldots,x_n$ free.

7.15 LEMMA. $A$ is a Horn formula iff $A$ can be written in the form $Q_1 \lor \ldots \lor Q_n$ where at most one of the $Q_i$ is atomic and the other disjuncts are negations of atomic formulas.

7.16 LEMMA. Let $T$ be a theory such that every direct product of models of $T$ is a model of $T$. Assume that

$$T \models \neg A_1 \lor \ldots \lor \neg A_n \lor B_1 \lor \ldots \lor B_m$$

where the $A_p$ and $B_q$ are atomic and $m>0$. Then there is an $i$, $1 \leq i \leq m$, such that

$$T \models \neg A_i \lor \neg B_j$$

7.17 THEOREM. A theory $T$ is equivalent to a theory whose nonlogical axioms are Horn formulas iff every submodel of a model of $T$ is a model of $T$ and every direct product of models of $T$ is a model of $T$.

PROOF:
The first half of the conclusion follows by Łos-Tarski's theorem. For the second half, let $(\mathcal{M}_i)_{i \in I}$ be a nonempty family of models of $T$ and let $\mathcal{M}$ be their direct product. Suppose $\mathcal{M} \models T$. Then $T$ has some Horn formula as a theorem such that

$$\forall x_1 \ldots \forall x_n \ B(x_1,\ldots,x_n)$$

where $B(x_1,\ldots,x_n)$ is a Horn formula having only $x_1,\ldots,x_n$ free.
(7-7) \[ M \models \neg A_1 \lor \ldots \lor \neg A_n \]
if no disjunct is atomic, and

(7-8) \[ M \models \neg A_1 \lor \ldots \lor \neg A_n \lor B \]
if one disjunct is atomic. Here we consider (7-8). The treatment of (7-7) is similar. From (7-8) follows that there are \( a_1, \ldots, a_k \in M \) such that

(7-9) \[ M \models A_j \implies (a_1, \ldots, a_k) \]
\[ M \models A_j \implies B \implies (a_1, \ldots, a_k) \]

By Lemma 7.13, there is a \( j \) such that

(7-10) \[ M_j \models B \implies (a_1, \ldots, a_k) \]

From Lemma 7.13 and (7-9), it follows for the same \( j \) that

(7-11) \[ M_j \models A_j \implies (a_1, \ldots, a_k) \]
\[ M_j \models A_i \implies (a_1, \ldots, a_k) \]

Then from (7-10) and (7-11),

(7-12) \[ M_j \models \neg A_1 \lor \ldots \lor \neg A_n \lor B \]

and hence \( M_j \models T \), contrary to the choice of the family \( \{ M_j \}_{j \in \mathbb{N}} \).

By Łos-Tarski's theorem, \( T \) is equivalent to an open theory \( T' \). Let \( A \) be a nonlogical axiom in \( T' \). Then \( A \) is a truth-function of atomic formulas. Therefore \( A \) can be written in conjunctive normal form \( C_1 \land \ldots \land C_k \) where each \( C_j \) is a disjunction of negated and unnegated atomic formulas. Obtain \( T'' \) from \( T' \) by replacing each such axiom \( A \) in \( T' \) by the \( k \) axioms \( C_1, \ldots, C_k \). Then \( T'' \) is an open theory and all nonlogical axioms are of the form

(7-13) \[ \neg A_1 \lor \ldots \lor \neg A_n \lor B_1 \lor \ldots \lor B_m \]

We now obtain \( T''' \) from \( T'' \) by replacing each nonlogical axiom in \( T'' \) by a new axiom as follows. If \( m = 0 \), let the corresponding axiom in \( T''' \) be the same. If \( m > 0 \), then by Lemma 7.16 there is a \( j \in \mathbb{N} \) such that

\[ T''' \models \neg A_1 \lor \ldots \lor \neg A_n \lor B_j \]

Replace Axiom (7-13) in \( T'' \) by

(7-14) \[ \neg A_1 \lor \ldots \lor \neg A_n \lor B_j \]

By sentential logic it is clear that \( T, T', T'', T''' \) are equivalent and that all nonlogical axioms of \( T''' \) are Horn formulas.

7.18 COROLLARY. Let \( S \) be a set of sentences in standard form and \( S^* \) its corresponding set of Skolem forms. Then \( S \) can be represented by a set of premises in HCL or Prolog iff every direct product of models of \( T(S^*) \) is a model of \( T(S^*) \).

PROOF:

\( S \) and \( S^* \) are not in general equivalent; but Theorem 2.20 shows that \( T(S^*) \) is a conservative extension of \( T(S) \). Therefore \( T(S) \) and \( T(S^*) \) give the same answers to questions formulated in the language \( L(S) \) of \( S \). It therefore suffices to prove that

(7-15) \[ T(S^*) \] is equivalent to a theory whose nonlogical axioms are Horn formulas iff every direct product of models of \( T(S^*) \) is a model of \( T(S^*) \).

This follows immediately from the theorem, from Łos-Tarski's theorem and the fact that since the nonlogical axioms of \( T(S^*) \) contain no existential quantifiers, \( T(S^*) \) is trivially equivalent with an open theory.

7.19 REMARK. The results in Section 2 and the proof of Theorems 7.17 show that \( T(S^*) \) is always equivalent to a theory whose nonlogical axioms are of the form (7-13), i.e., the axioms are general clauses. It follows that every problem can be represented and solved in GCL by the direct methods studied in sections 2-5. On the other hand, it is not all open theories \( T \) that have the property that the direct product of any family of models of \( T \) is a model of \( T \). It follows that HCL and Prolog are proper sublogics of GCL. There are problems which cannot be represented in HCL and Prolog by direct methods (though they can by the indirect methods of Theorem 6.37).

7.20 EXAMPLE. We give an example of a sentence which cannot be represented by a set of Horn sentences. When we formalize the sentences in Example 4.31 in clause form, we get three Horn sentences and the following non-Horn sentence

\[ S: \forall x (M(x) \land N(x) \rightarrow R(x, x) \lor R(0, x)) \]

Clearly, \( S^* = S \). We show that \( T(S^*) \) has two models \( A \) and \( B \) such that the direct product \( A \times B \) is not a model of \( T(S^*) \). Define

\[ A = \{ a_1, a_2 \}, \quad M = N = \{ a_1 \}, \quad R = \{ (a_2, a_1) \}, \quad b = a_2 \]
\[ \mathcal{B}(B, M, N, R, b) \]
\[ B = \{b_1, b_2\}, \quad M = N = \{b_1\}, \quad R = \{\langle b_1, b_1 \rangle \}, \quad b = b_2 \]

Then

\[ (7-16) \quad \mathcal{A} \models \forall x (M(x) \land N(x) \rightarrow R(x,x) \lor R(b,x)) \]

By Definition 7.9, the direct product of \( \mathcal{A} \) and \( \mathcal{B} \) is

\[ \mathcal{A} \times \mathcal{B} = (\mathcal{A} \times \mathcal{B}, M, N, R, b) \]
\[ M = N = \{(a_1, b_1)\}, \quad R = \{(a_2, b_1), (a_1, b_1)\}, \quad b = (a_2, b_2) \]

Since

\[ \mathcal{A} \times \mathcal{B} \models M(a_1, b_1) \land N(a_2, b_1) \]
\[ \mathcal{A} \times \mathcal{B} \models R(a_1, b_1), R(a_2, b_2) \land R(a_2, b_2) \]

it follows that

\[ (7-17) \quad \mathcal{A} \times \mathcal{B} \models \forall x (M(x) \land N(x) \rightarrow R(x,x) \lor R(b,x)) \]

By Corollary 7.18, it follows from (7-16) and (7-17) that the information in \( S \) cannot be adequately represented by any set of Horn clauses no matter how much logical ingenuity we apply.

In Example 2.38, we introduced the method of negative predicates. Using this method, a partial representation of sentences (1) and (2) in Example 4.31 can be given using only Horn clauses. That some information is lost by this procedure can be seen as follows. The set of sentences \( \{(1), (2)\} \) is inconsistent, but the set consisting of the four Horn clauses with the negative predicate is consistent. The details of the verification of this claim is left to the reader.

7.21 Remark. Horn formulas were first defined and studied by Horn (1951). He had no thought of any applications in programming and AI. His purpose was the investigation of model theoretic and logical properties of the very natural algebraic notion of a direct product of models. For this purpose he found Horn formulas useful. The importance of Horn clauses in logic programming was discovered much later by Colmerauer and Kowalski among others.

Many other results than Theorem 7.17 can be proved on Horn formulas, Horn sentences, and direct products of models. Chang and Keisler (1973) contains a comprehensive survey of these results.

Notes

This article is a survey of the logical foundations of logic programming, especially programming in Prolog. It is, with a paraphrase of Kowalski’s words, logic for machine problem solving. Much of the material has appeared before in research articles and in books like Kowalski (1979), Genesereth and Nilsson (1987), Johansson et al. (1989), Lloyd (1987), and Deville (1990); but the way of systematising and exposing the material is partly new, and there are many new examples. Notably, it is a systematisation which takes logic as its starting point rather than computing science as is the case in the named books. All proofs have been constructed from scratch and may also contain some novelties. The present survey is to some extent based on Chapter 13 of my Swedish language textbook Grundläggande logik [Fundamentals of Logic] (1994). I am indebted to the publisher, Studentlitteratur, Lund, for permission to use the material here.

It is assumed that the reader is acquainted with disjunctive and conjunctive normal form and prefix normal form. For deductions in the standard form, I use the system PD in Hansen (1992, 1994); but any other system for natural deduction in classical predicate logic will do as well. In all cases, Hansen (1994) is an adequate reference. Those interested in exercises in clause logic are referred to Chapter 13 of the same book.

References


Index

Absolute set theory 104
Abstraction principle 98
Ackermann, W. 47
Adams, E. 45
Algorithm 215, 232-233, 236
Alphabet 236
Analysis 58-59, 61-62, 67-68
Analytic philosophy 76-77
Analytic truth 85, 86
Anderson, A. 47-49
Aposteriori 85
A Priori 85, 87
Arithmetic theory 54-69, 138-141
Artificial intelligence 41, 223-224
Automaton 26, 36
Axiomatics 64
Axiom of choice 91
Axiom of infinity 91
Axiom of insight 62-67, 71
Backward resolution 177-178, 216
Barwise, J. 109
Bell, J. L. 104, 110, 127
Bello, N. 47-49
Birkhoff, G. 83
Birkhoff, G. 143, 146
Bottom-up resolution 177
Bracket term 107
Brouwer-first 203-205, 218, 223

Cantor, G. 98, 101
Carnap, R. 94-95
Cartesian question 84, 89
Category theory 70-71
Chang, C. C. 105, 250
Characteristic function 237
Christensen, N. E. 45
Church, A. 235
Church-Turing thesis 235
Classical semantics 145, 146
Clause 154
Clause form 154
Cognitive 84, 90
Cohen, D. 143, 145
Colossouer, A. 153, 250
Compatibility 142, 145
Complete arithmetic 56
Completeness 228, 231