MATHEMATICAL LOGIC

YIANNIS N. MOSCHOVAKIS

Glossary

Church-Turing Thesis: Claim that every computable function can be computed by a Turing machine.

Computability theory: Study of computable functions on the natural numbers.

Continuum hypothesis: Conjecture that there are only two sizes of infinite sets of real numbers.

Database: Finite, typically relational structure.

First order logic: Mathematical model of the part of language built up from the propositional connectives and the quantifiers.

Incompleteness phenomenon: Gödel's discovery, that sufficiently strong axiomatic theories cannot decide all propositions which they can express.

Model theory: Study of formal definability in first order structures.

Paradox: Counterintuitive truth.

Peano arithmetic: Axiomatic theory of natural numbers.

Proof theory: Study of inference in formal systems independently of their interpretation.

Propositional connectives: The linguistic constructs "and", "not", "or" and "implies".

Quantifiers: The linguistic constructs "there exists" and "for all".

Turing machine: Mathematical model of computing device with unbounded memory.

Unsolvable problem: A problem whose solution requires a non-existent algorithm.

Narrowly construed, mathematical logic is the study of *definition* and *inference* in mathematical models of fragments of language, especially the *first order logic* fragment. Logic has made critical contributions to the foundations of science, especially through the work of Kurt Gödel, and it also has numerous applications. For *set theory* and *theoretical computer science*, these applications are so important, that parts of these fields are normally included in the modern, broad conception of the discipline.

I. Propositional Logic, $\mathbb{PL}$

Each *logic* $\mathbb{L}$ has a *syntax* which delineates the grammatically correct linguistic expressions of $\mathbb{L}$, a *semantics* which assigns meaning to the correct expressions, and a structured *system of proofs* which specifies the rules by which some $\mathbb{L}$-expressions can be inferred from others.

There are other words to describe these things: *formal language* is sometimes used to describe a plain syntax, *formal system* often identifies a syntax together with an inference system (but without an interpretation), and *abstract logic* has been used to refer to a syntax together with an interpretation, leaving inference aside. It is, however, a fundamental feature of logic that it draws clean distinctions and studies the connections among these three aspects of language. We explain them first in the simplest example of the "logic of propositions", which is part of many important logics.

A. **Propositional Syntax**

The *symbols* of $\mathbb{PL}$ are the *connectives*

$$\neg \text{ (not)} \quad \& \text{ (and)} \quad \vee \text{ (or)}$$
$$\rightarrow \text{ (implies, if-then)}$$

the two parentheses '(', ')', and an infinite list of (formal) propositional variables $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2, \ldots$ which intuitively stand for declarative propositions, things like 'John loves Mary' or '3 is a prime number'. It has only one category of grammatically correct expressions, the *formulas*, which are *strings* (finite sequences) of symbols defined inductively by the following conditions:

F1. Each P_i is a formula.

F2. If A and B are formulas, then so are the expressions

$$\neg A \quad (A \,\&\, B) \quad (A \vee B) \quad (A \to B)$$

For example, if P and Q are propositional variables, then $(P \to Q)$ and $(P \vee \neg P)$ are formulas, which we read as "if P then Q" and "either P or not P".

The inductive definition gives a precise specification of exactly which strings of symbols are formulas, and also insures that each formula is either *prime*, i.e., just a variable P_i, or it can be constructed in exactly one way from its simpler *immediate parts*, by one of the connectives. This makes it possible to prove properties of formulas and to define operations on them by *structural induction* on their definition.

More propositional connectives can be introduced as "abbreviations" of formula combinations, e.g.,

$$A \leftrightarrow B \equiv ((A \to B) \,\&\, (B \to A))$$
$$A \vee B \vee C \equiv (A \vee (B \vee C)).$$

B. **Propositional Semantics**

If B stands for some *true* proposition, then $\neg B$ is false, independently of the "meaning" or internal structure of B. This is an instance of a general *Compositionality Principle for* $\mathbb{PL}$: *The truth value of a formula depends only on the truth values of its immediate parts.* The semantics of $\mathbb{PL}$ comprise the rules for computing truth values, and they can be summarized in Table 1, where 1 stands for 'truth' and 0 for 'falsity'. By the first line of this table, for

A	B	$\neg A$	$(A \,\&\, B)$	$(A \vee B)$	$(A \to B)$
1	1	0	1	1	1
1	0	0	0	1	0
0	1	1	0	1	1
0	0	1	0	0	1

TABLE 1. Truth value semantics.

example, if A and B are both true, then $\neg A$ is false while $(A \,\&\, B)$, $(A \vee B)$ and $(A \to B)$ are all true. Notice that if A is false, then $(A \to B)$ is reckoned to be true no matter what the truth value of B, so that 'if the moon is made of cheese, then $1 + 1 = 5$' is true (on the plausible assumption that the moon is not made of cheese). This *material implication* assumed by Propositional Logic has been attacked as counterintuitive, but it agrees with mathematical practice and it is the only useful interpretation of implication which accords with the Compositionality Principle.

Using these rules, we can construct for each formula A a *truth table* which tabulates its truth value under all assignments of truth values to the variables. For example, the truth table for $(Q \to P)$ consists of the first three columns of Table 2 while the first two

P	Q	$(Q \to P)$	$(P \to (Q \to P))$
1	1	1	1
1	0	1	1
0	0	1	1
0	1	0	1

TABLE 2.

and the last column give the truth table for $(P \to (Q \to P))$.

If n variables occur in a formula A, then the truth table for A has 2^n rows and determines an n-ary *bit function* v_A, with arguments and values in the two-element set $\{1, 0\}$. By the *Definitional Completeness Theorem*, *every n-ary bit function is v_A for some A*, so that the formulas of $\mathbb{PL}$ provide definitions (or "symbolic representations") for all bit functions.

A formula A is a *semantic consequence of* a set of formulas T (or T-*valid*) if every assignment to the variables which *satisfies* (makes true) all the formulas in T also satisfies A,. We write

$$T \models A \Leftrightarrow A \text{ is } T\text{-valid},$$

and $\models A$, in the important special case when T is empty, in which case A is called a *tautology*. A formula A is *satisfiable* if some assignment satisfies it, i.e., if $\neg A$ is not a tautology. Let

$$A \sim B \Leftrightarrow \{A\} \models B \text{ and } \{B\} \models A$$
$$\Leftrightarrow \models A \leftrightarrow B,$$

and call A and B *equivalent* if $A \sim B$.

Equivalent formulas define the same bit function, and they can be substituted for each other without changing truth values. Clearly

$$(A \to B) \sim (\neg A \vee B),$$

so that the implication connective is superfluous. In fact, every formula is equivalent to one in *disjunctive normal form*, i.e., a disjunction $A_1 \vee \cdots \vee A_k$ where each A_i is a conjunction of variables or negations of variables (*literals*).

C. Applications to Circuits

Each formula A with n variables can be realized by a *switching circuit* $C(A)$ with n *inputs* and one *output*, so that $C(\mathsf{P}_i)$ consists of just one input-output edge, $C(A\&B)$ is constructed by joining $C(A)$ and $C(B)$ with an *and-gate*, etc. Figure 1 exhibits the

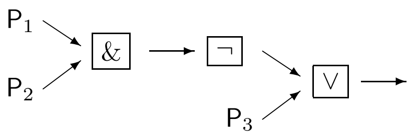


FIG. 1. The circuit for $(\neg((\mathsf{P}_1 \,\&\, \mathsf{P}_2) \vee \mathsf{P}_3)$.

circuit for $((\mathsf{P}_1 \,\&\, \mathsf{P}_2) \to \mathsf{P}_3)$ using the equivalent formula without implications, so that only $\neg$-, &- and $\vee$-gates are required. These are restricted circuits, of *fan-in* (maximum number of edges into a node) 2 and *fan-out* 1, but the Definitional Completeness Theorem implies that every n-ary bit function can be computed by some *formula circuit* $C(A)$.

There are basically two useful measures of circuit complexity, and both of them are faithfully mirrored in formulas. The *number of gates* of $C(A)$ is exactly the number of connectives in A and measures *size complexity* (construction cost), while the *depth* of $C(A)$, which measures the *time complexity* of computation, is exactly the *rank* of A, defined inductively so that $\mathrm{rk}(\mathsf{P}_i) = 1$, $\mathrm{rk}(A \,\&\, B) = \max(\mathrm{rk}(A), \mathrm{rk}(B)) + 1$ and similarly for the other connectives. One can now use natural manipulations of formulas to construct circuits which compute a given bit function with minimum size or time complexity, or to establish optimality results for the computation of bit functions by appealing to the formula representations of the circuits which realize them. For example, using disjunctive normal forms, one sees immediately that (if we do not care about cost), *every n-ary bit function can be computed by an unbounded fan-in circuit in no more than* 3 *time units*. There is, in general, a substantial *trade-off* between the size and time complexity of the circuits which compute a given bit function.

D. The Satisfiability Problem

The assertion that "$C(A)$ and $C(B)$ never give the same output on the same inputs" means precisely that "$(A \leftrightarrow \neg B)$ is a tautology", so that to detect that A and B do not have this *safety property* we need to determine whether the formula $\neg(A \leftrightarrow \neg B)$ is *satisfiable*.

Because of such natural formulations of "error detection" for circuits relative to given specifications, it is very important to find efficient algorithms for determining whether a given formula is satisfiable. The problem is of *non-deterministically polynomial time complexity* (*NP*), because it can be resolved by guessing ("non-deterministically") some assignment and then verifying that it satisfies A in a number of steps which is bounded by a polynomial in the length of A; and it is *NP*-complete, i.e., every *NP*-problem can be "reduced" to it by a polynomial reduction. This is a basic result of S. Cook, who introduced the complexity class *NP*, showed that it contains a large number of important problems, and asked if it coincides with the (seemingly) smaller class P of "feasible", *deterministically polynomial time* problems. The question whether $P = NP$ is the fundamental open problem of complexity theory; it amounts simply to the question whether the satisfiability problem can be solved by a deterministic, polynomial algorithm.

E. Propositional Inference

A *proof* of a formula A from a set of hypotheses T is any finite sequence

$$A_0, A_1, \ldots, A_{n-1}, A$$

which ends with A, and such that each A_i is either in T, or a $\mathbb{PL}$-*axiom*, or follows from previously listed formulas by a *rule of inference*. To make this notion precise we need to specify a set of $\mathbb{PL}$-axioms and rules of inference; and for these to be useful, it should be that they are few and easy to understand, and that the formulas provable from T are exactly the T-tautologies.

We need just one, binary *inference rule*:

$$\frac{A \qquad (A \to B)}{B} \text{ (Modus Ponens)}$$

This is *sound*, i.e., $\{A, (A \to B)\} \models B$, so that if A and $(A \to B)$ are both T-tautologies, then so is B.

An axiom is any instance of the following *axiom schemes*, where A, B and C are arbitrary formulas and we have omitted several parentheses which pedantry would require:

(1) $A \to (B \to A)$
(2) $(A \to B)$
$\qquad \to \Big((A \to (B \to C)) \to (A \to C)\Big)$
(3) $A \to (B \to (A \,\&\, B))$
(4) $(A \,\&\, B) \to A$ $\quad$ (4′) $(A \,\&\, B) \to B$
(5) $A \to (A \vee B)$ $\quad$ (5′) $B \to (A \vee B)$
(6) $(A \to C)$
$\qquad \to \Big((B \to C) \to ((A \vee B) \to C)\Big)$
(7) $(A \to B) \to \Big((A \to \neg B) \to \neg A\Big)$
(8) $\neg\neg A \to A$

These are all tautologies, and so every formula provable from T is T-valid. We write

$$T \vdash A \Leftrightarrow \text{there is a proof of } A \text{ from } T,$$

and it is not hard now to establish the basic

Soundness and Completeness Theorem for $\mathbb{PL}$. *For all sets* T *and any* A,

$$T \models A \Leftrightarrow T \vdash A.$$

F. **Boolean Algebras**

A *Boolean algebra* is a set $\mathcal{B}$ with at least two, distinct elements 0 and 1, a unary *complementation operation* $'$, and binary *infimum* $\cap$ and *supremum* $\cup$ operations such that certain properties hold. The standard example is the set $\mathcal{P}(M)$ of all subsets of some non-empty set M, with $0 = \emptyset$, $1 = M$ and the usual complementation, intersection and union operations, which for a singleton M gives the two-element set $\{1, 0\}$ of truth values; but there are others, e.g., the set of all finite and co-finite subsets of some infinite set, the set of all "closed and open" subsets of a topological space, etc.

Each formula A with n variables defines an n-ary function on every Boolean algebra $\mathcal{B}$, simply by letting the propositional variables range over $\mathcal{B}$ and replacing $\neg$, & and $\vee$ and $\to$ by $'$, $\cap$, $\cup$ and $\Rightarrow$ respectively, where

$$x \Rightarrow y = x' \cup y$$

on $\mathcal{B}$. Now the axioms for a Boolean algebra insure that *every propositional axiom defines a function with constant value* 1—in fact the particular choice of axiomatization for Boolean algebras (and there are many) is quite irrelevant as long as this fact obtains; and then the Completeness Theorem implies that *two formulas A and B define the same n-ary operation on all Boolean algebras exactly when* $A \sim B$, i.e., when A and B define the same bit function.

Boolean algebras have many important applications in mathematics (to measure theory, among other things), and they are the subject of the classical *Stone Representation Theorem* which identifies them all (up to isomorphism) with subalgebras of powerset algebras. In logic they are mostly used through the "non-standard" *Boolean semantics* of this subsection, which extend to richer logics and provide a powerful tool for *independence* (unprovability) results.

II. First Order Logic, $\mathbb{FOL}$

Consider the claim:

> If everybody has a mother, and
> every mother loves her children,
> then everybody is loved by
> somebody.

It is certainly true, it has the "linguistic form" of many similar (more substantial) claims in mathematics, and it appears to be *true by virtue of its form* and not because of any special properties of the words "mother", "love", etc. First Order Logic makes it possible to express complex assertions of this type and to show that they are true *by logic alone*. The symbolic expression of this one will be

$$\Big[(\forall x)(\exists y)M(x,y) \\ \& \ (\forall x)(\forall y)[M(x,y) \to L(y,x)]\Big] \\ \to (\forall x)(\exists y)L(y,x),$$

give-or-take a few parentheses and brackets which will be required to make the syntax completely precise.

A. **First Order Syntax**

The symbols of $\mathbb{FOL}$ are the propositional connectives, the parentheses, the *quantifiers*

$$\forall \text{ (for all)} \qquad \exists \text{ (there exists)}$$

the comma ',', the identity symbol '=', an infinite list $\mathsf{v}_0, \mathsf{v}_1, \ldots$ of *individual variables* which will denote arbitrary objects in some domain, and for each $n = 0, 1, \ldots$, two infinite lists of *function* and *relational symbols*

$$\mathsf{f}^n_0, \mathsf{f}^n_1, \ldots, \qquad \mathsf{P}^n_0, \mathsf{P}^n_1, \ldots,$$

which will stand for n-ary functions and relations on the objects.

There are two categories of grammatically correct expressions in $\mathbb{FOL}$, *terms* and *formulas*, defined recursively by the following conditions.

T1. Each variable v_i is a term.

T2. If $t_1, \ldots t_n$ are terms, then (the string) $\mathsf{f}^n_i(t_1, \ldots, t_n)$ is also a term. When $n = 0$, we write simply f^0_i.

F1. If $t_1, \ldots, t_n$ are terms, then the expressions

$$t_1 = t_2 \quad \mathsf{P}^n_i(t_1, \ldots, t_n)$$

are formulas, the latter written simply P_i when $n = 0$.

F2. If A and B are formulas, then so are the expressions

$$\neg A \quad (A \,\&\, B) \quad (A \vee B) \quad (A \to B)$$

F3. If A is a formula, then so are the expressions

$$(\forall \mathsf{v}_i)A \quad (\exists \mathsf{v}_i)A$$

Notice that by the notational convention in F1, all $\mathbb{PL}$-formulas are also $\mathbb{FOL}$-formulas.

This logic is called *first order* because quantification is only allowed over individuals; if we add formula formation rules

$$(\forall \mathsf{P}^n_i)A \quad (\exists \mathsf{P}^n_i)A$$

we obtain the formulas of *second order logic*, $\mathbb{SOL}$.

Consider the simple formula

$$(1) \qquad (\exists \mathsf{v}_2)(\neg \mathsf{v}_2 = \mathsf{v}_1 \,\&\, \mathsf{P}^1_1(\mathsf{v}_2)).$$

Its "translation" into English by the reading of the symbols we have introduced is

some object other than v_1
has the property P^1_1

which is exactly how we would translate the result of substituting v_3 for v_2 in it,

$$(\exists \mathsf{v}_3)(\neg \mathsf{v}_3 = \mathsf{v}_1 \,\&\, \mathsf{P}^1_1(\mathsf{v}_3)).$$

This is because both occurrences of v_2 in (1) are *bound* by the quantifier $\exists \mathsf{v}_2$, just as the occurrences of x are bound by the dx in $\int_0^1 x^2 dx$ and can be replaced by y without changing the meaning of the definite integral. On the other hand, the occurrence of v_1 in (1) is *free*, because it is not within the scope of any quantifier, and so the interpretation of v_1 clearly affects the meaning of (1).

Using the same simple example, consider the results of substituting $\mathsf{f}^1_0(\mathsf{v}_3)$ and $\mathsf{f}^1_0(\mathsf{v}_2)$ for v_1 in (1),

$$(\exists \mathsf{v}_2)(\neg \mathsf{v}_2 = \mathsf{f}^1_0(\mathsf{v}_3) \,\&\, \mathsf{P}^1_1(\mathsf{v}_2)),$$
$$(\exists \mathsf{v}_2)(\neg \mathsf{v}_2 = \mathsf{f}^1_0(\mathsf{v}_2) \,\&\, \mathsf{P}^1_1(\mathsf{v}_2)).$$

The first of these says of $\mathsf{f}^1_0(\mathsf{v}_3)$ what (1) says of v_2, but the second says that "something is not a fixed point of f^1_0 and has property P^1_1", which is quite different—evidently because the variable v_2 in $\mathsf{f}^1_0(\mathsf{v}_2)$ is "caught" by the quantifier $\exists \mathsf{v}_2$. The first is a *free substitution* (causing no confusion) while the second is not. We will denote the result of substituting the term t for the free occurrences of the variable x in some formula A by

$$A\{x :\equiv t\}$$

and we will tacitly assume that all substitutions are free.

Formulas of $\mathbb{FOL}$ are too messy to write down, and so we often resort to "informal descriptions" of them like the example about mothers loving their children above, recipes,

$$
\begin{aligned}
\imath \models t_1 = t_2 &\Leftrightarrow \imath(t_1) = \imath(t_2)\\
\imath \models \mathsf{P}_i^n(t_1,\dots,t_n) &\Leftrightarrow (\imath(\mathsf{P}_i^n))(\imath(t_1),\dots,\imath(t_n))\\
\imath \models \neg A &\Leftrightarrow \imath \not\models A\\
\imath \models (A \,\&\, B) &\Leftrightarrow \imath \models A \text{ and } \imath \models B\\
\imath \models (A \vee B) &\Leftrightarrow \imath \models A \text{ or } \imath \models B\\
\imath \models (A \to B) &\Leftrightarrow \imath \not\models A \text{ or } \imath \models B\\
\imath \models (\forall \mathsf{v}_i)A &\Leftrightarrow \text{for all } d \text{ in D,}\\
&\qquad \imath\{\mathsf{v}_i := d\} \models A\\
\imath \models (\exists \mathsf{v}_i)A &\Leftrightarrow \text{for some } d \text{ in D,}\\
&\qquad \imath\{\mathsf{v}_i := d\} \models A
\end{aligned}
$$

TABLE 3. The Tarski truth conditions.

really, from which the full, grammatically correct formula could (in principle) be constructed.

B. **First Order Semantics**

Whether (1) is true or false depends on the object v_1, on the function f_0^1, on the property P_1^1, and (most significantly) on the range of objects over which we interpret the existential quantifier—where do we search for things which may or may not satisfy P_1^1?

To interpret the formulas of $\mathbb{FOL}$ we must be given a *domain* D and an *interpretation* $\imath$, a function which assigns an object $\imath(\mathsf{v}_i)$ in D to each individual variable, an n-ary function $\imath(\mathsf{f}_i^n)$ on D to each n-ary function symbol f_i^n, and an n-ary relation $\imath(\mathsf{P}_i^n)$ on D to each P_i^n. Using these, first we extend inductively $\imath$ to all terms by

$$\imath(\mathsf{f}_i^n(t_1,\dots,t_n)) = (\imath(\mathsf{f}_i^n))(\imath(t_1),\dots,\imath(t_n)),$$

so that $\imath(t)$ is some object in D. To assign truth values to formulas, define first, for each variable x and d in D, the *update*

$$\jmath = \imath\{x := d\},$$

which agrees with $\imath$ on all function and relation symbols, and also on all individual variables, except that $\jmath(x) = d$. With the help of this basic operation, we can state in Table 3 the classical *Tarski truth conditions* which determine the truth of formulas relative to a fixed domain D and an interpretation $\imath$. The *truth value* of a formula A relative to an interpretation $\imath$ is 1 if $\imath \models A$ and 0 otherwise, and the *Compositionality Principle* extends to $\mathbb{FOL}$ in a straightforward manner and implies the following basic fact: *the truth value of A relative to $\imath$ depends only on the values of $\imath$ on the function and relation symbols which occur in A, and on the values $\imath(x)$ for the individual variables* which occur free *in A*.

The Tarski conditions do nothing more than translate formulas into English, in effect identifying $\mathbb{FOL}$ with a precisely formulated, small but very expressive fragment of natural language.

C. **Structures**

A *vocabulary* (or *signature*) is any finite sequence $\sigma = \{\mathsf{f}_1,\dots,\mathsf{f}_k,\mathsf{P}_1,\dots,\mathsf{P}_l\}$ of function and relation symbols, and $\mathbb{FOL}(\sigma)$ is the part of $\mathbb{FOL}$ whose formulas involve only the function and relation symbols of σ. The idea is to think of $\mathsf{f}_1,\dots,\mathsf{f}_k$ and $\mathsf{P}_1,\dots,\mathsf{P}_l$ as constants, denoting fixed functions and relations on some set D, and to use the formulas of $\mathbb{FOL}(\sigma)$ to study definability in *structures*

$$M = (D_M, f_1,\dots,f_k,P_1,\dots,P_l)$$

of vocabulary σ, where the *universe* D_M of M is any non-empty set, and $f_1,\dots,f_k$, $P_1,\dots,P_l$ are functions and relations which can be assigned to the vocabulary symbols, e.g., such that f_i is n-ary if f_i is n-ary.

An *M-assignment* is any function α from the variables to D_M, and it extends naturally to an interpretation α_M by the association of f_i with f_i and P_i with P_i; the standard notation for *structure satisfaction* is

$$M, \alpha \models A \Leftrightarrow \alpha_M \models A.$$

Formulas of $\mathbb{FOL}(\sigma)$ with no free variables are called *sentences* and (by the Compositionality Principle) they are simply true or false in every σ-structure, without reference to any assignment. They define properties of structures. We write

$$
\begin{aligned}
M \models A \Leftrightarrow &\text{ for any (and hence all) } \alpha,\\
&\quad M, \alpha \models A \quad (A \text{ a sentence}),
\end{aligned}
$$

and if $M \models A$, we say that M *satisfies* A or *is a model of* A.

While sentences define properties of structures, formulas with free variables can be used to define relations on structures. If, for example, A has at most one free variable x, we set

$$R_A(d) \Leftrightarrow M, \alpha\{x := d\} \models A,$$

where α is any assignment, since its only relevant value is updated in this definition. In the same way, formulas with n free variables define n-ary *relations* on σ-structures, the *first order definable* relations of M. A function $f : D_M^n \to D_M$ is first order definable if its graph

$$G_f(x_1, \ldots, x_n, w) \Leftrightarrow w = f(x_1, \ldots, x_n)$$

is first order definable. Some examples:

A *directed graph* is a structure $G = (D, E)$, where E is a binary "edge" relation on the set of "nodes" G, and it is a *graph* (undirected) if it satisfies the sentence

$$(\forall x)(\forall y)[E(x,y) \to E(y,x)].$$

Complete graphs (cliques) are characterized by the sentence

$$(\forall x)(\forall y)E(x,y),$$

while "diameter ≤ 2" is defined by

$$\begin{aligned}(\forall x)(\forall y)[x = y \vee E(x,y) \\ \vee (\exists z)[E(x,z) \mathbin{\&} E(z,y)]].\end{aligned}$$

Finite directed and undirected graphs are used to model many notions in computer science, e.g., circuits.

A *semigroup* (monoid) with identity is a structure $(S, e, \cdot)$ where the identity e is some specified member of S, $\cdot$ is a binary "multiplication" on S, and the following sentences are true:

$$\begin{gathered}(\forall x)(\forall y)[x \cdot (y \cdot z) = (x \cdot y) \cdot z], \\ (\forall x)(x \cdot e = x \mathbin{\&} e \cdot x = x).\end{gathered}$$

Here and in the sequel we write $t_1 \cdot t_2$ rather than the pedantically correct $\cdot(t_1, t_2)$.

In addition to semigroups, there are *groups*, *rings*, *fields* and *ordered fields*, *vector spaces*, and any number of other structures which are the stuff of "abstract" algebra. These classes of structures are all characterized by first order axioms, and the use of methods from logic is becoming increasingly important in their study.

Two structures M_1 and M_2 are *isomorphic* if some one-to-one correspondence between their universes carries the functions and relations of M_1 to those of M_2. *Isomorphic structures satisfy the same first order sentences*, but the converse is not true, as we will see in II-F.

D. **Databases**

In the most general terms, a *database* is just a finite structure, typically *relational*, i.e., without functions, only relations. "Finite" does not mean "small" or "simple", and in the interesting applications databases are huge structures of large and complex vocabularies, with basic relations such as "x is an employee born in year n", "y is the supervisor of x", etc. Properties of structures are usually called *queries* in database theory, and one of the main tasks in the field is to develop representations for databases which support fast algorithms for *updating*, entering new information in the base and *data testing*, determining the truth or falsity of queries. As it happens, both *updating and data testing are very efficient for first order queries*, and so database systems, including the industry standard SQL make heavy use of methods from first order logic.

Motivated by Database Theory, a good deal of research has been done since the 1970s in *Finite Model Theory*, the mathematical and logical study of finite structures. For a rather surprising, basic result, let

$$\begin{aligned}&\mathrm{Prob}_\sigma[M \models A : |D_M| = n] \\ &\quad = \text{the proportion of } \sigma\text{-structures} \\ &\qquad\qquad \text{of size } n \text{ which satisfy} A,\end{aligned}$$

where structures are counted "up to isomorphism".

The $\mathbb{FOL}$ 0-1 *Law. For each sentence A of $\mathbb{FOL}(\sigma)$ in a relational vocabulary, either*

$$\lim_{n\to\infty} \mathrm{Prob}_\sigma[M \models A : |D_M| = n] = 1,$$

or

$$\lim_{n\to\infty} \mathrm{Prob}_\sigma[M \models A : |D_M| = n] = 0,$$

i.e., either A or $\neg A$ is *asymptotically true.*

More advanced work in this area is concerned primarily with the algorithmic analysis of queries on finite structures, especially in logics richer than $\mathbb{FOL}$.

E. **Arithmetic**

Most basic is the structure of arithmetic

$$N = (\mathbb{N}, 0, 1, +, \cdot),$$

where $\mathbb{N} = \{0, 1, \ldots\}$ is the set of (non-negative) *natural numbers* and $+$ and $\cdot$ are the operations of addition and multiplication. The first order definable relations and functions on N are called *arithmetical*, and they obviously include addition, multiplication and the ordering on $\mathbb{N}$, which is defined by the formula

$$x \leq y \equiv (\exists z)[x + z = y].$$

By a basic Lemma of Gödel, if a function f is determined from arithmetical functions g and h by the equations

$$(2) \qquad \begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y+1, \vec{x}) = h(f(y, \vec{x}), y, \vec{x}), \end{cases}$$

then f is also arithmetical. Thus *exponentiation* x^y is arithmetical, with $g(x) = 1$, $h(w, y, x) = w \cdot x$, and, with some work, so is the function $p(x)$ which enumerates the prime numbers,

$$p(0) = 2,\ p(1) = 3,\ p(2) = 5,\ \ldots.$$

In fact, the scheme of *Primitive Recursion* (2) is the basic method by which functions are introduced in number theory, so that, with some work, all fundamental number theoretic relations and functions are arithmetical, and all celebrated theorems and open problems of the theory of numbers are expressed by first order sentences of N. These include the *Prime Number Theorem*, *Fermat's Last* (Wiles') *Theorem*, and the (still open) question whether *there exist infinitely many twin pairs of prime numbers.*

F. **Model Theory**

The mathematical theory of structures starts with the following basic result:

Compactness and Skolem-Löwenheim Theorem. If every finite subset of a set of sentences T has a model, then T has a countable model.

For an impressive application, let (in the vocabulary of arithmetic)

$$\Delta_0 \equiv 0, \quad \Delta_{m+1} \equiv (\Delta_m + 1),$$

so that the *numeral* Δ_m is about the simplest term which denotes the number m, add a constant c to the language, and let

$$T = \{A : N \models A\} \\ \cup \{\Delta_0 \leq \mathsf{c}, \Delta_1 \leq \mathsf{c}, \Delta_2 \leq \mathsf{c}, \ldots\}.$$

Every finite subset S of T has a model, namely

$$N_S = (\mathbb{N}, 0, 1, +, \cdot, m),$$

where the object m which interprets c is some number bigger than all the numerals which occur in formulas of S. So T has a countable model

$$N_T = (\overline{\mathbb{N}}, \overline{0}, \overline{1}, \overline{+}, \bar{\cdot}, c),$$

and then $\overline{N} = (\overline{\mathbb{N}}, \overline{0}, \overline{1}, \overline{+}, \bar{\cdot})$ is a structure for the vocabulary of arithmetic which satisfies all the first order sentences true in the "standard" structure N *but is not isomorphic with* N—because it has in it some object c which is "larger" than all the interpretations of the numerals $\Delta_0.\Delta_1, \ldots$. It follows that, with all its expressiveness, First Order Logic does not capture the isomorphism type of complex structures such as N.

These *non-standard* models of arithmetic were constructed by Skolem in the 30s. Later, in the 50s, Abraham Robinson constructed by the same methods *non-standard models of analysis*, and provided firm foundations for the classical Calculus of Leibnitz with its infinitesimals and "infinitely large" real numbers.

Model Theory has advanced immensely since the early work of Tarski, Abraham Robinson and Malcev. Especially with the contributions of Shelah in the 70s and, more recently, Hrushovsky, it has become one of the most mathematically sophisticated branches of logic, with substantial applications to algebra and number theory.

G. **First Order Inference**

The proof system of First Order Logic is an extension of that for Propositional Logic, first by *identity axioms* which insure that $=$ is an equivalence relation and a *congruence* for all function and relation symbols, e.g., for unary function symbols,

$$(\forall x)(\forall y)[x = y \rightarrow f(x) = f(y)].$$

In addition, there are two axioms for the quantifiers,

$$A\{x :\equiv t\} \rightarrow (\exists x)A \quad (\forall x)A \rightarrow A\{x :\equiv t\},$$

assuming that the term substitutions are free; and there are two new inference rules,

$$\frac{C \rightarrow A}{C \rightarrow (\forall x)A} \qquad \frac{A \rightarrow C}{(\exists x)A \rightarrow C}$$

which can be used only when the variable x is not free in C. Proofs from a set T of $\mathbb{FOL}(\sigma)$ sentences are defined exactly as for $\mathbb{PL}$, and we set again

$$T \vdash A \Leftrightarrow \text{there is a proof of } A \text{ from } T.$$

Notice that without the restriction on the quantifier rules, the sequence

$$P(x) \rightarrow P(x), P(x) \rightarrow (\forall x)P(x), \\ (\exists x)P(x) \rightarrow (\forall x)P(x)$$

would be a proof of $(\exists x)P(x) \rightarrow (\forall x)P(x)$, which is, obviously, not valid. With the restriction, however, for every structure M, if every M-assignment satisfies the hypothesis of either new rule, then every M-assignment satisfies the conclusion, so that the quantifier inference rules are *sound*.

H. **Gödel's Completeness Theorem**

A *model* of a set of sentences T in $\mathbb{FOL}(\sigma)$ is any structure M which satisfies every A in T, in symbols

$$M \models T \Leftrightarrow \text{for all } A \text{ in } T, M \models A.$$

We also write

$$T \models A \Leftrightarrow \text{for all } M, \\ M \models T \implies M \models A,$$

which extends to $\mathbb{FOL}(\sigma)$ the *semantic consequence* relation of $\mathbb{PL}$. From the comments above:

Soundness Theorem for $\mathbb{FOL}$. *If* $T \vdash A$, *then* $T \models A$.

The fundamental fact about First Order Logic is the converse of this result:

Completeness of $\mathbb{FOL}$. *If* $T \models A$, *then* $T \vdash A$.

It may be argued that the semantic consequence relation $T \models A$ captures the intuitive notion A *follows from the assumptions in* T *by logic alone*, in the sense that it insures that A is true whenever all the hypotheses in T are true, independently of the meaning of the function and relation symbols. Granting that and considering the strong expressibility of First Order Logic discussed in II-C above, we may then argue further that the Completeness Theorem answers definitively (for science) the ancient question of *what follows from what by logic alone*: a proposition A follows from certain assumptions T as a matter of logic (and independently of the facts), if A and T can all be expressed faithfully as $\mathbb{FOL}(\sigma)$ assertions about some σ-structure M, and $T \vdash A$. On this view, it is hard to overemphasize the importance of this result for the foundations of mathematics and science.

Incidentally, there is an obvious extension of the Tarski conditions to Second Order Logic, e.g.,

$$\imath \models (\forall \mathsf{P}_i^n)A \;\Leftrightarrow\; \text{for all } n\text{-ary } P \text{ on } D, \\ \imath\{\mathsf{P}_i^n := P\} \models A.$$

However, there is no useful Completeness Theorem for $\mathbb{SOL}$, as we will see in IV-F.

I. **Proof Theory**

If Model Theory is the study of semantics independently of inference, then Proof Theory can be viewed as the mathematical investigation of formal proofs independently of interpretation. This has always been one of the most active research areas of logic, and it has been invigorated in recent years by its substantial applications to computer science, including *automated deduction*, an important component of *artificial intelligence*. Key to these applications—and the basic result of Proof Theory—is the *Extended Normal Form Theorem* of Gentzen, whose somewhat weaker (but simpler) Herbrand version is fairly easy to describe.

There are four Herbrand inference rules, and they apply to n-ary disjunctions

$$A_1 \vee \cdots \vee A_n.$$

Two of them are *structural*, and they clearly preserve meaning: you can interchange the order of the disjuncts, or delete one of two occurrences of the same disjunct. The other two are *quantifier rules*,

$$\frac{A_1 \vee \cdots \vee A_n\{x :\equiv t\}}{A_1 \vee \cdots \vee (\exists x)A_n} \quad \frac{A_1 \vee \cdots \vee A_n}{A_1 \vee \cdots \vee (\forall x)A_n}\ *$$

where the $*$ indicates that the $\forall$-rule can only be used if the variable x is not free in its conclusion. The result applies only to sentences without identity and in *prenex normal form*, i.e., looking like

$$(\mathsf{Q}_1 x_1) \cdots (\mathsf{Q}_n)B$$

where each Q_i is $\forall$ or $\exists$ and B is quantifier-free.

Herbrand's Theorem. Every provable $=$-free sentence A of $\mathbb{FOL}(\sigma)$ in prenex form can be derived from a provable quantifier-free disjunction by the four Herbrand rules.

The restriction to prenex sentences is not essential, because every formula can be converted to an equivalent prenex one by the application of simple rules which can be added to the system.

The theorem asserts (in part) that every provable sentence A has a "normal" proof, in which only formulas of "quantifier rank" no greater than A occur. This is a powerful tool for proof-theoretic studies. As for applications, all automated deduction systems use Herbrand-like inference systems (or their Gentzen variants), and the programming language PROLOG is based entirely on this idea.

The proof of Herbrand's Theorem is constructive: an algorithm is defined, which computes for each proof Π of a prenex sentence A a Herbrand proof Π', and then it is shown by simple, combinatorial arguments that Π', indeed, proves A. The additional, effective content is significant for the foundational applications of the theorem (for example to consistency proofs), and also in the applications to automated deduction.

It should be emphasized that the simplistic slogans "Model Theory = no inference" and "Proof Theory = no semantics" are often honored in the breach: like the Completeness Theorem, most fundamental results of logic are about connections between truth and proof, and some of the deepest results in one part of the discipline depend on methods and ideas from the other.

III. Gödel's Incompleteness Theorem

Having established that $\mathbb{FOL}$ proves all *logical truths*, it is natural to ask if it can also prove—from some natural set of axioms—all *mathematical truths*. This is not possible, by Gödel's fundamental result, whose special case for *arithmetical truths* we discuss in this section.

A. The Incompleteness of Peano Arithmetic

The classical *Peano axioms* for arithmetic comprise the properties of the successor

$$(3) \quad x + 1 \neq 0 \qquad x + 1 = y + 1 \rightarrow x = y,$$

the recursive definitions of addition and multiplication,

$$(4) \quad \begin{cases} x + 0 = x \\ x + (y + 1) = (x + y) + 1, \end{cases}$$

$$(5) \quad \begin{cases} x \cdot 0 = 0 \\ x \cdot (y + 1) = x \cdot y + x, \end{cases}$$

and the *Induction Axiom* which cannot be expressed fully in First Order Logic. Its Second Order Logic version is

$$(\forall P)\Big[\Big(P(0) \,\&\, (\forall x)(P(x) \rightarrow P(x+1))\Big) \rightarrow (\forall x)P(x)\Big],$$

and the best we can do in $\mathbb{FOL}$ is to adopt the Axiom Scheme

$$(6) \quad \Big(A\{y :\equiv 0\} \,\&\, (\forall x)(A\{y :\equiv x\} \rightarrow A\{y :\equiv x+1\})\Big) \rightarrow (\forall x)A\{y :\equiv x\}.$$

The set PA of (first order) Peano axioms is obtained by taking the correctly spelled versions of all the formulas in (3)–(6) and adding enough universal quantifiers in front of them so that they become sentences. This is a very strong set of axioms, it can prove all simple properties of numbers and most of their deep properties too—although proving a theorem from PA is harder than proving it using, say, methods from analysis, and number theorists distinguish and value "elementary proofs" in PA.

Gödel's First Incompleteness Theorem. There is a sentence $\mathbf{g}$ *in* $\mathbb{FOL}(0, 1, +, \cdot)$, *such that* $N \models \mathbf{g}$ *but* $\mathsf{PA} \not\vdash \mathbf{g}$.

One's first thought is that we can overcome this "incompleteness phenomenon" by strengthening PA, perhaps add Gödel's own $\mathbf{g}$ to it, or use the Second Order Logic version of the Induction Axiom along with a suitable axiomatization of Second Order Logic. None of this helps: Gödel's fundamental discovery is that first order truth in N (and every other sufficiently rich structure) simply cannot be presented usefully as an "axiomatic theory". We will make this precise in a more general version of the Incompleteness Theorem in the next section.

B. **Coding (Gödel numbering)**

The basic ingredients of the proof of the Incompleteness Theorem are *coding* and *self-reference.*

In analytic geometry we "code" (represent) points in the plane by pairs of real numbers, their coordinates, so we can translate geometrical questions into algebraic problems and solve them by calculation. Gödel's basic idea is to code the syntactic objects of $\mathbb{FOL}(0, 1, +, \cdot)$—terms, formulas, proofs—by natural numbers, so that their properties are translated into properties of numbers, which can then be expressed in $\mathbb{FOL}(0, 1, +, \cdot)$ and (perhaps) proved in PA.

Since all syntactic objects are strings of symbols, if we view a proof $A_1, \dots, A_{n-1}$ as a sequence of formulas separated by commas, it is enough to code strings, and we can do this in (at least) one simple minded way: we enumerate the symbols of the language

$\neg$	$\&$	$\vee$	$\rightarrow$	(	)	$\forall$	$\exists$	,	=	0	1	+	$\cdot$	v_0	v_1	$\cdots$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\cdots$

and we set

$$[a_0 a_1 a_2 \cdots a_m] = 2^{n_0} 3^{n_1} 5^{n_2} \cdots p(m)^{n_m},$$

where n_i is the code of the symbol a_i and $p(i)$ is the i'th prime number. For example, the (correctly spelled) prime formula

$$+(\mathsf{v}_1, 0) = \mathsf{v}_0$$

has the horrendously large code

$$2^{13} 3^5 5^{16} 7^9 11^{11} 13^6 17^{10} 19^{15}.$$

The size of codes is irrelevant: what matters is that every string of symbols (and hence every term, formula and proof) has a code from which it can be reconstructed, by the *Unique Factorization Theorem* for numbers; and (more significantly) that PA is powerful enough to express and prove simple properties of formulas and proofs, thus translated into properties of numbers. For example, if Δ_n is the numeral denoting n, as above, then PA can prove all true, basic relations among numerals, e.g.,

$$m + n = k \implies \mathsf{PA} \vdash \Delta_m + \Delta_n = \Delta_k.$$

Less trivially, the basic (coded) *proof relation*

$\mathrm{Proof}_{\mathsf{PA}}(a, p) \Leftrightarrow$

> a is the code of some sentence A and p is the code of a proof of A from PA

is defined by some formula $\mathsf{Proof}_{\mathsf{PA}}$ with v_1 and v_2 free, and PA can prove its basic properties, e.g.,

$$\mathrm{Proof}_{\mathsf{PA}}(a, p) \implies \mathsf{PA} \vdash \mathsf{Proof}_{\mathsf{PA}}\{\mathsf{v}_1 :\equiv \Delta_a, \mathsf{v}_2 :\equiv \Delta_p\}.$$

Similarly, the relation

$D(a, p) \Leftrightarrow$

> a is the code of some formula A with only v_1 free, and p is the code of a PA-proof of $A\{\mathsf{v}_1 :\equiv \Delta_a\}$

is defined by some formula D with just $\mathsf{v}_1, \mathsf{v}_2$ free. Set

$$A \equiv (\forall \mathsf{v}_2)\neg\mathsf{D}$$

so that only v_1 is free in A, and if a is the code of A, set

$$\mathbf{g} \equiv A\{\mathsf{v}_1 :\equiv \Delta_a\}.$$

Unscrambling the definitions, $\mathbf{g}$ asserts that *there is no* PA*-proof of* $A\{\mathsf{v}_1 :\equiv \Delta_a\}$; but $\mathbf{g}$ *is* $A\{\mathsf{v}_1 :\equiv \Delta_a\}$, so that $\mathbf{g}$ claims its own unprovability; and a careful analysis of the situation shows that, indeed, $\mathbf{g}$ cannot be provable in PA, else PA would prove a contradiction. This also shows, that **g** is true.

It is not that simple, of course, and much delicate analysis and computation must be done to establish that $D(a,p)$ is arithmetical and to derive a formal contradiction from the assumption that **g** is PA-provable. Key to the proof is the "self-reference" in the definition of $D(a,p)$, which uses the coding, and the argument depends on the strength (not the weakness) of the axiomatic system PA. Coding and self-reference have become standard tools of logic since Gödel's work, and they have found substantial applications in many areas, including computer science and set theory.

IV. Computability

It is easy to determine whether an arbitrary equation $a_0+a_1x+\cdots+a_nx^n=0$ with integer coefficients $a_0,\dots,a_n$ has integer solutions, since every integer root must divide a_0, and so all we have to do is to test the finitely many divisors of a_0. The problem is not so easy for equations in k unknowns

$$\sum_{r_1+\cdots+r_k\le n} a_{r_1,\dots,r_n} x_1^{r_1}x_2^{r_2}\cdots x_k^{r_k}=0, \tag{7}$$

and it is much more interesting, in fact

> *to find an algorithm which determines whether* (7) *has a solution*

is No. 10 in David Hilbert's famous 1900 list of 23 open problems in mathematics. Diophantine equations are notoriously difficult to solve, and one might suspect that no algorithm would do the job, but how can you prove such an assertion? Using ideas and techniques from Gödel's work and motivated by questions arising from it, logicians developed in the 30s a tool for establishing *absolute unsolvability results* of this kind which led to some spectacular applications, including a rigorous proof of the unsolvability of Hilbert's 10th.

The most direct approach was by Turing, who reasoned that algorithms should be implemented by "mechanical devices" and introduced "abstract machines" that can perform symbolic computations some ten years before digital computers were invented.

A. **Turing machines**

A Turing machine M is determined by a finite alphabet $S_M=\{s_0,\dots,s_k\}$, a finite set $Q_M=\{q_0,\dots,q_m\}$ of (internal) *states*, and a finite table of *transitions* of the form

$$q,s \;\mapsto\; q',s',m$$

where q,q' are states, s,s' are in S_M or the special "blank" symbol $\sqcup$, and *the move* m is -1, 0 or $+1$. No two transitions are *activated* by the same pair q,s on the left. We imagine that, at any moment, M is in some internal state q and sits in front of an infinite "tape" with symbols in some of its *cells*. The machine can only "see" the symbol s just in front of it, and does nothing (*halts*) unless one of its transitions is activated by the pair q,s; in which case it switches to state q', it replaces s by s' on the tape, and it moves

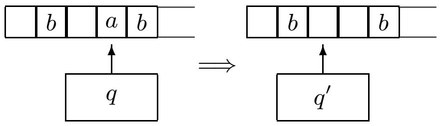
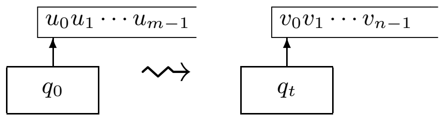


Fig. 2. $q,a \;\mapsto\; q',\sqcup,-1$.

left (if it can), right or none-at-all, depending on whether m is -1, $+1$ or 0.

A machine M starts computing facing the leftmost cell, with an arbitrary string *input*



$u=u_0\cdots u_{m-1}$ on the tape, and it may *diverge* (never halt), for example if $u=11$ and M has the two transitions

$$q_0,1 \;\mapsto\; q_0,1,+1 \qquad q_0,\sqcup \;\mapsto\; q_0,1,+1$$

If it halts, then its *output* on u is the string $M[u]=v_0\cdots v_{n-1}$ at the left end of the tape, until the first blank (and it is possible that $M[u]$ is empty.)

Finally, M *computes* a string function $f:S_1^*\to S_2^*$ if $S_1\cup S_2\subseteq S_M$ and for every string $u\in S_1^*$, $M[u]=f(u)$. By identifying each natural number n with the string $||\cdots|$ of $n+1$ *tallies* from the one-member alphabet $\{|\}$ (unary notation), the notion covers functions whose arguments or values are either strings or numbers. Moreover, if we code strings by numbers as above, then the transformation $u \mapsto [u]$ and its inverse can be computed by a Turing machine, so that a string function is Turing computable exactly when its "coded version" is computable, and we can safely confuse the two notions.

B. **The Church-Turing Thesis**

Turing argued persuasively that the symbolic computations of any "finite mechanical device" with access to unbounded memory can be simulated by one of his machines, and he has been fully justified by the subsequent developments in computers. Church had already made an equivalent (though less well justified) claim, and so the new fundamental principle carries both famous names:

The Church-Turing Thesis: *A string function $f : S_1^* \to S_2^*$ is computable if and only if it can be computed by a Turing machine M on some alphabet $S_M \supseteq S_1 \cup S_2$.*

The Church-Turing Thesis cannot be rigorously proved, as it identifies the intuitive, informal notion of "computability" with the precise, mathematical property of *Turing computability*. Within mathematics, it is officially a definition, much like the definitions of *arclength* or *area* in terms of integrals. But mathematical definitions are not entirely arbitrary: when we "define" the length of the circumference of a circle of radius r by an integral which computes out to $2\pi r$, we fully expect that if we draw such a circle and measure its circumference, it will turn out to be $2\pi r$, within the margin of error of our measurements. Similarly, when we prove that a certain string function f is not Turing computable, we fully expect that nobody will ever discover an algorithm which computes f, because no such algorithm exists. This is the standard method of application of the Thesis.

Evidence for the Church-Turing Thesis comes from Turing's analysis, from the sixty-odd years of failed attempts to contradict it, and from the robustness of the notion of Turing computability. Many classes of functions were defined in the thirties claiming to capture the notion of "computable" from different perspectives, including Church's λ-*definable functions*, Post's *canonical systems*, the *general recursive functions* of Gödel, Herbrand and Kleene, Kleene's μ-*recursive functions* and, in the forties, Markov's (formal) *algorithms*; each of these was proved equivalent to Turing computability, and the "simulation techniques" developed for these proofs make it seem very unlikely that some algorithm will ever be discovered which cannot be simulated by a Turing machine.

It should be emphasized, however, that the Church-Turing Thesis does not provide a rigorous definition for the notion of *algorithm*, which remains informal. Complexity results about algorithms are rigorously grounded on various so-called *computation models* which embody diverse features of actual computers. When we simulate these models by Turing machines, the time and space complexity of computations increase substantially, and so we cannot claim that the informal algorithm has been faithfully modeled. On the other hand, the time complexity increase is bound by a polynomial factor for all the known simulations, so that the class P of polynomial problems can be defined in terms of Turing machines without ambiguity.

Turing-computable functions are also called *recursive*, because of the basic Gödel-Herbrand-Kleene characterization mentioned above.

C. **Unsolvable Problems**

A set of strings (or *problem*) $Q \subseteq S^*$ from a finite "alphabet" S is *computable* (*recursive*, *solvable*, *decidable*) if some Turing machine M computes its *characteristic function*

$$c_Q(u) = \begin{cases} 1 & \text{if } u \in Q, \\ 0 & \text{otherwise,} \end{cases}$$

otherwise it is *unsolvable* or *undecidable*. The definitions apply to problems about natural numbers, coded in unary; to problems about $\mathbb{FOL}$-formulas, by identifying (for example) each variable v_i by a similar sequence $\mathsf{vv}\cdots\mathsf{v}$ of $i+1$ v's, so that the syntax of $\mathbb{FOL}$ is based on a finite vocabulary; and to relations (sets of n-tuples) on strings or numbers, by thinking of $u_1,\ldots,u_n$ as a single string.

Each Turing machine can be represented by a string of 0's and 1's which codes its alphabet, internal states and transitions, and this leads to the first and most basic unsolvability result, due to Turing:

The Halting problem: *It is undecidable whether an arbitrary Turing machine M halts on an arbitrary binary string u.*

For the proof, Turing constructed a *universal machine U* which can simulate every other, i.e.,

$$U[\overline{M},u] = M[u], \quad \text{if } \overline{M} \text{ is the code of} M.$$

This treatment of *programs* as *data* is, of course, routine today.

All unsolvability results are (ultimately) established by reducing the Halting Problem to them, i.e., showing that if such-and-such a function were computable, then the Halting Problem would be solvable. The proofs are often difficult and generally depend on results specific to the field in which the problem arises.

In mathematics, the problems which have been proved unsolvable include:

Hilbert's 10th: *Whether a given Diophantine equation has integer solutions* (Matijasevich, following work of Martin Davis, Hilary Putnam and Julia Robinson).

The Word Problem for Groups: *Whether two words denote the same element in a finitely generated, finitely presented group* (P. Novikov, W. Boone).

The Homeomorphism Problem for 4-*manifolds*: *Whether the orientable n-manifolds represented by two triangulations are homeomorphic*, for $n \geq 4$ (A. Markov). This problem is solvable for 2-manifolds, by their classical representation as spheres with handles, and it is still open for 3-manifolds, pending (among other things) the resolution of the Poincaré Conjecture.

There is also a large number of unsolvable problems in Computer Science.

D. **Undecidable Theories**

A *theory* T in $\mathbb{FOL}(\sigma)$ is any set of sentences closed under consequence,

$$T \vdash A \implies A \in T.$$

The two basic examples are theories of σ-structures

$$\mathrm{Th}(M) = \{A \mid M \models A\},$$

and *axiomatic theories* of the form

$$T = \mathrm{Th}(T_0) = \{A : T_0 \vdash A\},$$

where T_0 is a decidable *set of axioms* T_0. The terminology is natural, because we would certainly demand of any "axiomatization" that it can be decided effectively whether an arbitrary sentence is an axiom.

Every decidable theory T is axiomatizable since $\mathrm{Th}(T) = T$ when T is a theory, but the converse fails, in general, and in particular for $T_0 = \emptyset$ when the vocabulary is not trivial:

Church's Theorem: *If the vocabulary σ includes at least one binary function or relation symbol, then it is undecidable for a sentence A of $\mathbb{FOL}(\sigma)$ whether $\vdash A$.*

A $\mathbb{FOL}(\sigma)$-theory T is *consistent* if it does not contain a contradiction $A \,\&\, \neg A$, and it is *complete* if for every sentence A, either A or $\neg A$ is in T. It is easy to verify that *every consistent, axiomatizable, complete theory is decidable*, and we can use this to formulate and prove a very general version of the Gödel Incompleteness Theorem. The key tool is the notion of *translation*.

Suppose T_1 and T_2 are theories, perhaps in different vocabularies σ_1 and σ_2—e.g., T_1 might by $\mathrm{Th}(\mathsf{PA})$, and T_2 might be some axiomatic set theory. A *translation* of T_1 into T_2 is a computable string function ρ which assigns a sentence $\rho(A)$ of $\mathbb{FOL}(\sigma_2)$ to every sentence A of $\mathbb{FOL}(\sigma_1)$ and preserves propositional logic and T_1-inference, i.e.,

$$\begin{aligned}
T_2 &\vdash \rho(\neg A) \leftrightarrow \neg\rho(A)\\
T_2 &\vdash \rho(A \,\&\, B) \leftrightarrow \rho(A) \,\&\, \rho(B)\\
T_1 \vdash A &\implies T_2 \vdash \rho(A).
\end{aligned}$$

Notice that the identity function $\rho(A) = A$

translates every theory into itself.

The Gödel Incompleteness Theorem (Rosser's form). *If T is a consistent, axiomatizable theory and Peano arithmetic* $\mathrm{Th}(\mathsf{PA})$ *is translatable into T, then T is undecidable and hence incomplete.*

In short, every consistent axiomatic system in which a reasonable amount of mathematics can be developed is undecidable and incomplete.

To state the strongest corresponding result about theories of structures, we need the simple fact that *every computable set is arithmetical*, essentially due to Gödel.

Tarski's Theorem. If $\mathrm{Th}(N)$ *is translatable into* $\mathrm{Th}(M)$*, then* $\mathrm{Th}(M)$ *is not arithmetical*, a fortiori it is not decidable.

To apply Tarski's Theorem, you need (in effect) to give a first order definition of the natural numbers within the given structure. One of the first results of this type was *the undecidabilty of the theory of rational numbers* $\mathrm{Th}(\mathbb{Q}, 0, 1, +, \cdot)$ (Julia Robinson), but there are many others, and there are also many difficult open problems in this area.

On the other hand, many interesting theories are decidable, including the following:

- The theory $\mathrm{Th}(\mathbb{N}, 0, 1, +)$ of arithmetic without multiplication (Presburger).
- The theory $\mathrm{Th}(\mathbb{Q}, \leq)$. This coincides with the theory of every *dense, linear ordering without endpoints.*
- The theory $\mathrm{Th}(\mathbb{C}, 0, 1, +, \cdot)$ of the complex number field, which coincides with the theory of every algebraically closed field of characteristic 0 (Tarski, Abraham Robinson).
- The theory $\mathrm{Th}(\mathbb{R}, 0, 1, +, \cdot, \leq)$ of the ordered field of real numbers, which coincides with the theory of every real closed field (Tarski).

The classical result here is Tarski's decidability of the ordered field of real numbers, which (using coordinates) implies that *Euclidean geometry is decidable*, in a sense trivializing much of ancient Greek mathematics! It is still open whether the extended theory $\mathrm{Th}(\mathbb{R}, 0, 1, +, \cdot, \leq, \uparrow)$ (with $x \uparrow y = x^y$ for $x > 0$) is decidable, but there has been substantial progress in this problem with Wilkie's Theorem, that *every set in $\mathbb{R}$ which is first order definable using exponentials is a finite union of intervals.*

E. The Second Incompleteness Theorem

What sorts of true sentences are not provable in sufficiently strong axiomatizable theories? If $T = \mathrm{Th}(T_0)$ is axiomatizable in $\mathbb{FOL}(\sigma)$, then the (coded) proof relation

$\mathrm{Proof}_T(a, p) \Leftrightarrow$

> a is the code of some sentence
> A in $\mathbb{FOL}(\sigma)$ and p is the code
> of a proof of A from T

is Turing computable, and hence arithmetical. Using this, we can construct a sentence Concis_T in the vocabulary of PA which expresses naturally the consistency of T and establish the following:

Gödel's Second Incompleteness Theorem (Rosser's form). *If T is consistent, axiomatizable and ρ translates* $\mathrm{Th}(\mathsf{PA})$ *in T, then T cannot prove the translation* $\rho(\mathsf{Concis}_T)$ *of its consistency sentence.*

The theorem makes it clear that we cannot axiomatize a substantial part of mathematics *in any way whatsoever* so that the consistency of the system can be established "constructively": because the (presumably simple) "constructive methods" we would be willing to use in a consistency proof should be part of the "substantial part of mathematics" we want to axiomatize. Beyond its obvious foundational significance, the Second Incompleteness Theorem has numerous applications, especially in comparing the strength of various hypotheses in Axiomatic Set Theory.

F. Hierarchies

A set Q of strings or numbers is Σ^0_2 if

$$u \in Q \Leftrightarrow (\exists x_1)(\forall x_2) R(u, x_1, x_2),$$

where the quantified variables range over natural numbers and the *matrix* R is computable, and it is Π^0_3 if, for all u

$$u \in Q \Leftrightarrow (\forall x_1)(\exists x_2)(\forall x_3) R(u, x_1, x_2, x_3)$$

with the same restrictions. The definitions extend naturally to all k, and we also set

$$\Delta^0_k = \Sigma^0_k \cap \Pi^0_k.$$

Kleene, who introduced these classes, showed that

$$\Delta^0_1 = \text{the class of recursive sets},$$

$$\begin{array}{ccccccccc} & & \Sigma^0_1 & & & & \Sigma^0_2 & & \\ & \subsetneq & & \subsetneq & & \subsetneq & & \subsetneq & \\ \Delta^0_1 & & & & \Delta^0_2 & & & & \cdots \\ & \subsetneq & & \subsetneq & & \subsetneq & & \subsetneq & \\ & & \Pi^0_1 & & & & \Pi^0_2 & & \end{array}$$

and that a non-empty set Q is Σ^0_1 exactly when it is *recursively* (or *computably*) *enumerable*, i.e., if

$$Q = \{f(0), f(1), \dots\}$$

with some recursive $f : \mathbb{N} \to S^*$. Moreover, these classes increase properly and exhaust the arithmetical sets. A similar hierarchy

$$\Sigma^1_k,\ \Pi^1_k,\ \Delta^1_k$$

for the *analytical* (second-order definable) sets is constructed by allowing the quantified variables to range over the unary functions $\alpha : \mathbb{N} \to \mathbb{N}$ and the matrix to be arithmetical, so that all arithmetical sets are in Δ^1_1.

These hierarchies classify the analytical sets of natural numbers and strings by the logical complexity of their (simplest) definitions, and they are powerful tools in the theory of definability. For example,

every axiomatizable theory is Σ^0_1.

This rules out an axiomatization of Second Order Logic $\mathbb{SOL}$, whose set of valid sentences (on the empty vocabulary) is not analytical. Somewhat surprisingly, it also rules out an axiomatization of the theory

$$T_f = \{A \mid \text{ for all finite } (D,E), (D,E) \models A\}$$

of finite graphs, which is Π^0_1 but not Σ^0_1 (Trachtenbrot).

G. **Turing reducibility**

Imagine a Turing machine with a second *query tape* which it handles exactly like its primary tape, implementing somewhat more complex transitions of the form

$$q, s_1, s_2 \ \mapsto\ q', s'_1, s'_2, m_1, m_2$$

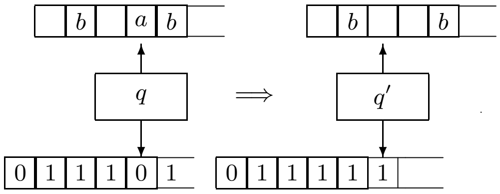


FIG. 3. $q, a, 0 \ \mapsto\ q', \sqcup, 1, -1, +1$

It also has a special *query state* $q_?$, and when it goes into $q_?$, the computation stops and does not resume until some external agent (the *oracle*) replaces the contents on the query tape by some string.

A string function f is *computable relative to some given* g if it can be computed by such an oracle machine, provided each time $q_?$ is reached, the string u on the query tape is replaced by the value $g(u)$. We let

$$f \leq_T g \Leftrightarrow f \textit{ is computable in } g,$$

and we extend this notion of *Turing reducibility* to sets of natural numbers via their characteristic functions.

It is not hard to show that *there exist Turing-incomparable sets of numbers* (Kleene-Post). In fact, *there exist Turing-incomparable recursively enumerable sets*, but this was quite hard to prove and it was a celebrated open question for some twelve years, known as *Post's Problem.* The simultaneous, independent discovery in 1956 by Friedberg and Muchnik of the *priority method* which proved it, initiated an intense study of Turing reducibility which is still, today, one of the most active research areas of logic, the largest (and technically most sophisticated) part of *computability* or *recursion theory.*

V. Recursion and Programming

In its most general form, a recursive definition of a function x is expressed by a *recursive* (or *fixed point*) *equation*

$$(8) \qquad x(t) = f(t, x),$$

where the *functional* $f(t,x)$ provides a method for computing each value $x(t)$, perhaps using ("calling") other values of x in the process. It is possible to characterize the computable functions on the natural numbers using simple recursive equations of this form, generalizations of the primitive recursive definition (2) in III-E. Though conceptually less direct than Turing's approach through idealized machines, this modeling of computability by "recursiveness" provides a powerful tool for establishing properties of computable functions, and it is especially useful in the theory of programming languages.

A. **Recursive equations**

Not every recursive equation (8) has a solution x, and some have many, e.g., the trivial $x(t) = x(t)$ which is satisfied by every function. The basic result which guarantees *canonical solutions* to a large class of recursive equations comes from the theory of *partially ordered sets*.

A *partially ordered set* or *poset* is a structure $(D, \leq_D)$, where $\leq$ is a binary relation and for all x, y, z in D,

$$x \leq_D x, \quad [x \leq_D y \,\&\, y \leq_D z] \implies x \leq_D z$$
$$[x \leq_D y \,\&\, y \leq_D x] \implies x = y;$$

a subset C of D is a *chain* if every two members of C are $\leq_D$-comparable, i.e., $x \leq_D y$ or $y \leq_D x$; and a poset D is *complete* if every chain in D has a *supremum* (least upper bound).

Every complete poset has a least element $\perp$ (the supremum of the empty chain!), and every set A can be turned into a *flat poset*

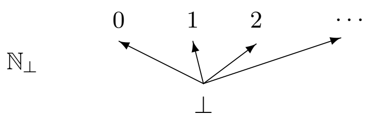


FIG. 4. Flat poset.

$A_\perp$ by adding a "bottom" below all its otherwise incomparable elements. Other, basic examples include the set of all subsets of a set A (under $\subseteq$) and the set of all (finite and infinite) sequences from some set, under "extension". The Cartesian product of complete posets is complete, and, more importantly, *if W is complete, then the function spaces of all arbitrary, monotone or Scott-continuous mappings $\pi : D \to W$ are also complete*, with the *pointwise partial ordering*

$$\pi \leq \rho \Leftrightarrow \text{for all } x, \pi(x) \leq \rho(x).$$

Here $\pi : D \to W$ is monotone if

$$x \leq_D y \implies \pi(x) \leq_W \pi(y),$$

and it is Scott-continuous if, in addition, for every chain C in D,

$$\pi(\text{supremum}(C)) = \text{supremum}(\pi[C]).$$

The Least-Fixed-Point Theorem. If $(D, \leq)$ is a complete poset and $\pi : D \to D$ is monotone, then the recursive equation

$$x = \pi(x) \tag{9}$$

has a least solution.

The theorem is proved by setting recursively

$$x^0 = \perp, \quad x^{n+1} = \pi(x^n). \tag{10}$$

In the simplest case, which is sufficient for the applications to programming languages, the mapping π is Scott continuous, and then

$$\overline{x} = \text{supremum}\{x^0, x^1, \ldots\}$$

is the least fixed point of π. For the full result we need to extend the iteration (10) into the "transfinite", using recursion on ordinal numbers.

There is a rich theory of complete posets and various kinds of mappings on them, mostly motivated by the applications to programming, but also by earlier work in *abstract recursion*, the generalization of computability to abstract structures.

B. **Programming Languages**

From the mathematical point of view, a programming language $\mathbb{P}$ is very much like a logic, with a syntax, a semantics, and an *implementation*, which plays the role of an inference system.

The *syntax* is generally much more complex than that of logics, with many different categories of grammatically correct expressions. There are variables of various kinds, some of them for *functions* of specified *types*; constants which are meant to denote *acts* of interaction with the environment (input, output, interrupts); and various ways of combining grammatically correct expressions to produce new ones, using *programming constructs* like composition, "while loops", functional abstraction and recursion. Some *closed* expressions (with no free variables) corresponding to the "sentences" of a logic are singled out, typically called *programs*. With all this complexity, the "grammar" is still specified by an induction, as it is for logics, so that it is again possible to prove properties of correct expressions and to define operations on them by *structural induction*.

In the *denotational semantics* introduced by Dana Scott, a programming language $\mathbb{P}$ is interpreted in a structure $(D, —)$ whose universe D is a complete poset, the *domain*. The points of D may include concrete *data* (words from some finite alphabet), but also functions of various sorts and complex mathematical structures which model computations, interactions, etc. For each correct expression A and each assignment α to the variables, the denotation $[\![A]\!](\alpha)$ is a point in D, determined by a structural induction of the following general form: first a (Scott-continuous) recursive equation (9) is constructed from α and the denotations of the parts of A, and then we take

$$[\![A]\!](\alpha) = \text{the least fixed point of} \, [x = \pi(x)].$$

The use of recursive equations is absolutely essential here, to interpret the iteration and recursive constructs which are at the heart of programming languages.

The *implementation* is a function which assigns to each program A a "machine" M_A—or, more concretely, code in the machine language of some processor—which *computes* the denotation $[\![A]\!]$ of A. In the simplest case, $[\![A]\!]$ might just be a sequence of external acts, like "printing" some file or drawing some picture on a monitor; more often $[\![A]\!]$ is a function relating input to output, or a "strategy" in some game, by which the machine responds to a sequence of external stimuli. As with inference systems, implementations come in a great variety of shapes and forms (*compilers* and *interpreters*, to name two), but they must have the basic *soundness property*, that M_A "computes" $[\![A]\!]$ in a well understood way which relates the abstract (mathematical) denotations of programs to the behavior of machines.

Even with this grossly oversimplified description, it should be clear that the basic methodology of logic—the clean distinction between *syntax*, *semantics* and *inference*—has had an immense influence on the development of programming languages; and that the fundamental, related notions of *symbolic computation* and *recursion* introduced by logicians in the 30s are essential to the understanding of programming languages.

In the other direction, the study of programming languages—spurred by the need for applications—has introduced a host of interesting problems in logic, chief among them the question of *logic of programs*: what are the natural formal languages and inference systems in which the fundamental properties of programs can be expressed and rigorously proved? Much work has been done on this, but it is fair to say that the question is still open, and a formidable challenge to logicians and computer scientists.

VI. Alternative Logics

From the many alternative logics which are obtained by changing the syntax, semantics or inference system of First Order Logic, we consider, very briefly, just two.

A. **Modal and Temporal Logic**

Modal Logic goes back to Aristotle, the traditional founder of logic, who took *necessity* as one of the basic linguistic constructs worthy of logical study. The modern syntax is obtained by adding to $\mathbb{FOL}$ the propositional *box operator* $\Box$, so that with each formula A we have the formula $\Box A$ (with the same free variables), read *necessarily* A. The *possibility* operator is defined by the abbreviation $\Diamond A \equiv \neg\Box\neg A$.

Modal formulas are interpreted in *Kripke structures*

$$M = (W, s_0, \{M_s \mid s \in W\}, R),$$

of a specified vocabulary σ, where W is some set of *possible worlds*; s_0 is a specified "actual world"; each M_s is a σ-structure associated with the world s; and $R(s,t)$ is an *accessibility relation* on the worlds, intuitively standing for "t is a possible alternative to s". There are no fixed, general assumptions about the accessibility relation or the interpretations of the given relations on the various worlds; it could be, for example, that "Mary is John's wife" in the actual world s_0, but in alternative possible worlds John's wife might be Ellen, John may not have a wife—or he may not even exist. Assignments associate objects in the possible worlds to individual variables, and the basic, semantic relation $M_s, \alpha \models A$ is defined by the Tarski conditions (for structures) with the additional clause

$$\begin{aligned} &M_s, \alpha \models \Box A \\ &\quad \Leftrightarrow \text{for all } t, \text{ if } R(s,t), \text{ then } M_t, \alpha \models A. \end{aligned}$$

For example, if R is *transitive*,

$$R(s,t) \,\&\, R(t,t') \implies R(s,t'),$$

then, the formula

$$\Box A \to \Box\Box A \tag{11}$$

is satisfied by all assignments, in all possible worlds, while it may fail for some A in non-transitive structures. Finally,

$$M, \alpha \models A \Leftrightarrow M_{s_0}, \alpha \models A.$$

Different conceptions of "necessity" can be modeled by placing appropriate restrictions on the accessibility relation, for example that it be transitive, linear, etc., and there is a question of constructing a suitable inference system and proving the appropriate Completeness Theorem in each case. A great deal of interesting work has been done in this area, much of it motivated by puzzles in the philosophy of language.

If we take $W = \mathbb{N}$ for the set of possible worlds, with $s_0 = 0$ and $R(s,t) \Leftrightarrow s \le t$, and if we read $\Box A$ as "*from now on* A", we get one version of Temporal Logic, very useful for applications to computing systems. The worlds are interpreted by the *states* of some finite state machine, the propositional variables stand for properties of states, and the propositional formulas (which suffice) can express interesting properties of the system, especially if we augment the language with some additional, natural primitives like Next with the truth condition

$$M_s \models \mathsf{Next}\, A \Leftrightarrow M_{s+1} \models A.$$

For example, $\Box(p \to \mathsf{Next}\, q)$ says that "every state which has property p is followed immediately by one which has property q", and $\Diamond\Box p$ says that "p will eventually become and remain true", both interesting properties of finite state machines. This *temporal logic is decidable*, and so are various extensions of it, in which essentially all interesting *liveliness* and *fairness* properties of finite state machines can be expressed, so that one can *mechanically verify* the "correct behavior" of finite state machines. The relevant algorithms are practical, if not simple, they are used commercially, and they provide a spectacular example of the emerging field of *applied logic*.

B. **Intuitionistic Logic**

First Order Intuitionistic Logic $\mathbb{FOL}_I$ has the same syntax as $\mathbb{FOL}$, and almost the same inference system: we simply replace the *Double Negation Law* $\neg\neg A \to A$, (8) in I-E, by the weaker

$(8)_I$ $\neg A \to (A \to B)$.

Kripke has established a Completeness Theorem for $\mathbb{FOL}_I$ using a variation of his semantics for Modal Logic, and this is useful for obtaining unprovability results for $\mathbb{FOL}_I$. The language, however, is meant to be understood constructively, and so it is not really possible to explain its semantics fully within classical mathematics. Aside from philosophical concerns, the real interest of Intuitionistic Logic comes from the proof theory of $\mathbb{FOL}_I$, which, somewhat surprisingly, also has important applications to Computer Science. Some sample results:

(1) For any two sentences A and B,

$$\vdash_I A \vee B \implies \vdash_I A \text{ or } \vdash_I B,$$

and hence $\not\vdash_I p \vee \neg p$.

(2) In *Heyting Arithmetic*, i.e., the axiom system PA of III-A with Intuitionistic Logic,

for any sentence $(\exists x)A$,

$$\begin{aligned}&\mathsf{PA} \vdash_I (\exists x)A \\ &\quad\Longrightarrow \text{ for some } n, \mathsf{PA} \vdash_I A\{x :\equiv \Delta_n\}.\end{aligned}$$

(3) If $\mathsf{PA} \vdash_I (\forall x)(\exists y)A$ and $(\forall x)(\exists y)A$ is a sentence (no free variables), then there is a computable function f, such that for all n, $\mathsf{PA} \vdash_I A\{x :\equiv \Delta_n, y :\equiv \Delta_{f(n)}\}$.

This last result is obtained with Kleene's *Realizability Theory*, and it illustrates the following general principle: *from a constructive proof of* $(\forall x)(\exists y)R(x,y)$*, we can extract an algorithm which computes for each* x*, some* y *such that* $R(x,y)$. There are obvious applications of this idea in Computer Science, and much of the current research in Intuitionism is motivated by it.

VII. Set Theory

Sets are collections into a whole of definite and separate objects of our intuition or thought, according to Georg Cantor, who initiated their mathematical study in the mid 1870s. Thus the basic relation of the theory is *membership* ($\in$),

$$x \in A \Leftrightarrow x \text{ is a member of } A,$$

and a set is completely determined by its members,

$$A = B \Leftrightarrow (\text{for all } x)[x \in A \Leftrightarrow x \in B].$$

Finite sets can be simply enumerated, e.g., $A = \{0, 5, 7\}$. Infinite sets are usually specified by means of some condition $P(x)$ which characterizes their members, and we write

$$A = \{x \mid P(x)\}$$

to indicate that A "is the set of all objects which satisfy $P(x)$".

Cantor was led to the study of arbitrary, *abstract sets* in his effort to understand the structure of some specific sets of real numbers or *pointsets*, and the theory which he created still exhibits today these two related but separate concerns. The *theory of pointsets* or *descriptive set theory* is primarily a theory of definability on the real numbers, and it is characterized by its applications to other fields of mathematics, especially analysis. *Abstract set theory* is primarily a theory of *counting*, an extension of combinatorics to the transfinite. The best set-theoretic results are about the interaction between these two poles of the subject.

At about the same time as Cantor's original contributions, Gottlob Frege initiated an effort to create a *foundation of mathematics* on the basis of set theory. Frege's approach was different (he took "function" rather than "set" as his primitive notion) and his original program was overly ambitious and failed. He had the right basic idea, however, that *all objects of classical mathematics can be "defined within set theory", so that their properties can be* (ultimately) *derived from properties of sets*. It took some time for this to take hold, but it is fair to say that since the 1930s, set theory has been the official language of mathematics, just as mathematics is the official language of science. This richness of the field makes it fertile ground for logical investigations, and it is not an accident that logicians have been involved with set theory from the beginning.

A. **Cardinal Arithmetic**

There are exactly as many left shoes in a (normal) shoe store as there are right shoes—and we can be sure of this without counting, because of the obvious one-to-one correspondence between left and right shoes. The principle here is that *equivalent sets have the same number of members*,

$$|A| = |B| \Leftrightarrow A \sim_c B, \tag{12}$$

where $A \sim_c B$ indicates that some one-to-one correspondence exists between the members of A and the members of B, and

$$|X| = \text{the number of objects in the set } X.$$

This is a basic tool in mathematics: we count a set A by establishing a one-to-one correspondence between its members and the members of some already-counted set B. Moreover, if we set

$$A \preccurlyeq_c B \Leftrightarrow \text{for some subset } C \subseteq B, A \sim_c C,$$

then, obviously,

$$|A| \le |B| \Leftrightarrow A \preccurlyeq_c B, \tag{13}$$

and we can often prove indirectly that *there are objects in* B *which are not in* A by showing (using arithmetic) that $|A| < |B|$, so that $B \subseteq A$ is impossible.

Cantor proposed to associate a *cardinal number* $|X|$ with every (finite or infinite) set X, so that (12) and (13) hold, and then to use similar counting and (infinite) *cardinal arithmetic* techniques in the study of arbitrary sets. One might expect problems, because a finite set cannot be equivalent with one of its proper subsets (by the so-called *Pigeonhole Principle*), while

$$\mathbb{N} = \{0, 1, 2, \ldots\} \sim_c \{0, 2, 4, \ldots\} \tag{14}$$

via the correspondence $f(n) = 2n$. Cantor showed that, despite this "paradox", his cardinal arithmetic is a powerful tool with important applications in almost all areas of mathematics.

Cantor's first fundamental discovery was that there are (at least) two infinite sizes of sets: if

$$\aleph_0 = |\mathbb{N}|, \quad \mathfrak{c} = |\mathbb{R}| = |\text{the real numbers}|$$

are the cardinal numbers of the two most basic sets in mathematics, then

$$\aleph_0 < \mathfrak{c}. \tag{15}$$

A set A is *countable* if $|A| \leq \aleph_0$, otherwise it is, like $\mathbb{R}$, *uncountable*.

To define the arithmetical operations on (possibly infinite) cardinal numbers, choose sets K, L with no members in common so that $\kappa = |K|$, $\lambda = |L|$, and set

$$\begin{aligned} \kappa + \lambda &= |K \cup L|, \\ \kappa \cdot \lambda &= K \times L, \\ \kappa^\lambda &= |(L \to K)|. \end{aligned}$$

Here the *union* $K \cup L$ is the set of all objects which belong to either K or L; the *Cartesian product* $K \times L$ is the set of all ordered pairs (x, y) with $x \in K$ and $y \in L$; and the *function space* $(L \to K)$ is the set of all functions $f : L \to K$. If κ and λ are finite, we get the usual sum, product and exponential, noting, in particular, that there are κ^λ functions from a set of size λ to one of size κ. Moreover, all the familiar arithmetical identities hold—e.g., addition and multiplication are associative and commutative, multiplication distributes over addition, $\kappa^0 = 1$, and

$$\kappa^{\lambda+\mu} = \kappa^\lambda \cdot \kappa^\mu, \quad \left(\kappa^\lambda\right)^\mu = \kappa^{\lambda\cdot\mu}.$$

As examples of "proofs by counting", Cantor showed first that

$$\aleph_0 + \aleph_0 = \aleph_0 \cdot \aleph_0 = \aleph_0 \tag{16}$$

(basically because of (14)), and

$$\mathfrak{c} = 2^{\aleph_0}.$$

Both of these facts are easy, but they support the computation

$$\mathfrak{c}^2 = 2^{\aleph_0} \cdot 2^{\aleph_0} = 2^{\aleph_0+\aleph_0} = 2^{\aleph_0} = \mathfrak{c},$$

which means that *there is a one-to-one correspondence between the line and the plane*—and, hence, between the line and real n-space, for every n! This was new, it was surprising, and it was proved by "plain arithmetic". Eventually it motivated the development of *dimension theory*, whose basic result is that *there is no continuous, one-to-one correspondence of real n-space with real m-space unless $n = m$.* Moreover, *the set of rational numbers is countable*, and so is the set of *algebraic numbers*, the solutions of polynomial equations

$$a_0 + a_1 x + a_2 x^2 + \cdots + x^n = 0$$

with integer coefficients. Thus, since $\mathbb{R}$ is uncountable, "by simple counting" *there exist transcendental* (not algebraic) *real numbers*, a famous result of Liouville's whose original proof had rested on delicate convergence arguments for infinite series. It was a "killer application" which made set theory instantly known (and somewhat notorious) in the mathematical community.

Cardinal addition and multiplication satisfy the following *absorption laws* which basically trivializes them in the infinite case:

$$\text{if } 0 < \kappa \leq \lambda \text{ and } \lambda \text{ is infinite,} \quad \text{then } \kappa + \lambda = \kappa \cdot \lambda = \lambda.$$

For exponentiation, however, Cantor extended (15) to the general inequality

$$\kappa < 2^\kappa,$$

which provides infinitely many distinct "orders of infinity", perhaps what people meant when they referred to *Cantor's Paradise*. Exponentiation is the source of the deepest questions about infinite sets, chief among them Cantor's *Generalized Continuum Hypothesis* (GCH), the claim that for all infinite κ,

$$\text{(GCH)}\quad 2^\kappa = \kappa^+ \\ = \text{the least cardinal number } > \kappa.$$

The "ordinary" case (CH) $2^{\aleph_0} = \aleph_0^+$ was No. 1 in Hilbert's list, it dominated set-theoretic research in the 20th century and, in a sense, it is still open today.

In addition to the cardinal numbers, which count the members of a set

one, two, three, ...

in the finite case, Cantor also introduced infinite versions of the *ordinal numbers*

first, second, third, ...

which assign position in a sequence. These are associated with "transfinite sequences", i.e., *well ordered structures* $(A, \leq)$, where $\leq$ is a *linear ordering* on A (so that $x \leq y$ or $y \leq x$, for all x, y in A) and *every non-empty subset of A has a least element.* Every ordinal number α has a *successor* $\alpha + 1$ which defines "the next position", and every set of ordinal numbers A has a *least upper bound* $\sup A$. The least infinite ordinal number ω defines the first position with in-

$$0,\ 1,\ 2,\ \ldots \omega,\ \omega+1,\ \omega+2,\ \ldots \\ \omega\cdot 2,\ \omega\cdot 2+1,\ \ldots$$

finitely many predecessors, and it is a *limit ordinal*, without an immediate predecessor.

Ordinal arithmetic has fewer direct applications than the arithmetic of cardinal numbers, but well ordered structures and ordinal numbers are the fundamental tools in the study of *transfinite iteration*, which is rich in applications. In a typical case, a function $f : A \to B$ is defined by *recursion* on some well ordered structure $(A, \leq)$, and then the crucial properties of f are established by *induction* along $\leq$. Moreover, the exact specification of the relation $\leq$ is often unimportant: all that matters is that *some* relation well orders A, in other words that A be *well orderable.*

(WOP) *Is every set well orderable?*

Specifically, is the set $\mathbb{R}$ of real numbers (where many of the applications lie) well orderable? The natural ordering of $\mathbb{R}$ won't do, since (for example) $\mathbb{R}$ has no least element, and it is hard to imagine how one could arrange all the real numbers into a transfinite sequence, with each point followed by its successor and every non-empty subset having a least element. The *Continuum Problem* (whether CH is true or not) and this *Well Ordering Problem* were the central open problems in set theory at the turn of the 20th century.

B. **The paradoxes**

Cantor developed his theory on the basis of the following *General Comprehension Principle* which flows naturally from his "definition" of sets quoted in the beginning of this section: *every definite* (unambiguous) *property $P(x)$ of mathematical objects, has an extension, the set $A = \{x \mid P(x)\}$* which "collects into a whole" all the objects which satisfy $P(x)$, so that

$$(17)\qquad x \in A \Leftrightarrow P(x).$$

But this is not generally true: because if

$$R = \{x \mid \text{ is a set and } x \notin x\},$$

then, from (17),

$$R \in R \Leftrightarrow R \text{ is a set and } R \notin R \Leftrightarrow R \notin R$$

which is absurd. The argument was discovered in 1902 by Bertrand Russell, and it was not the first contradiction in set theory. However, earlier "paradoxes" (some of them known to Cantor) were technical, not unlike the paradoxes with infinitesimals which had been commonplace in the Calculus some years earlier, and it was thought that they would go away in a careful development of the subject. The *Russell Paradox* is not technical, it goes to the heart of the nature of sets, and it threw the mathematical community into a spin.

L. E. J. Brouwer initiated the *intuitionistic program* which denies that abstract sets are meaningful objects of study and also rejects some of the basic principles of logic. Mathematical objects cannot be said to "exist" in any sense independent of (mental) "mathematical activity"; and to prove that *some x has property P*, one must *construct* some specific object x which has property P—it is not enough to derive a contradiction from the assumption that *no x has property P*. Intuitionism had a strong influence in the philosophy of mathematics and remains a vibrant field of study within logic, but it never carried much favor with mathematicians: too much of classical mathematics must be thrown out to satisfy its tenets.

Hilbert proposed to "save" classical mathematics from the paradoxes and Brouwer's attack by formalizing as large a part of it as possible in some first order, axiomatic theory T, and then establishing the consistency of T by absolutely safe, *finitistic* methods. *Formalism* is the reading of *Hilbert's Program* as a philosophical view: it alleges that once T is chosen, then T is all there is—there is nothing more to mathematics but the study of the inference relation $T \vdash A$, with no reference to meaning. Aside from the impact of Gödel's Second Incompleteness Theorem (IV-E) which weakens it, Formalism also fails to account for the applications of mathematics: it is hard to see how the existence or not of certain patterns of meaningless symbols can have any bearing on the escape velocity of a rocket.

From those reluctant to abandon the traditional, *realist* view that mathematical objects are, well, *real*, no matter how abstract and difficult to pin down, Russell first proposed to replace set theory by his famous *theory of types*: it is claimed (roughly, and in the later *simple* version due to Ramsey), that every mathematical object is of a certain (natural number) *type n*, and that every set A is of some successor type $n + 1$, such that the members of A are of the immediately preceding type n. Type theory is awkward to apply and it yields only a poor shadow of Cantor's set theory, albeit without the paradoxes. It never gained favor as a true alternative to set theory, although it has been studied extensively as a logical system, it has found its own applications (especially recently, to programming languages), and many of its fundamental ideas were eventually incorporated in the reformulation of set theory which eventually prevailed.

What has prevailed is *Axiomatic Set Theory*, first proposed in 1904 by Zermelo as a pragmatic way to avoid the paradoxes by rebuilding Cantor's set theory on the basis of a few set-theoretic principles which are basic, simple and well understood by their uses in classical mathematics. Formalists can accept it, as nothing more but the choice of a specific set of axioms, whose "truth" is irrelevant—if, at all, meaningful. But it is the realists who, in the end, have received the greatest comfort from axiomatic set theory: because the systematic development of consequences of the axioms eventually led to a narrower, more concrete concept of *set*, which ultimately justified the axioms.

Much of modern logic was created in response to the challenge of the set-theoretic paradoxes, and that is another reason why the discipline is so intimately tied with set theory.

C. **Zermelo-Fraenkel Set Theory**

There are eight axioms in ZFC (*Zermelo-Fraenkel Set Theory with Choice*), and it is assumed that they are interpreted over some given domain of *sets* $\mathbb{V}$, which comes endowed with a binary *membership* condition, $x \in y$. The formal theory ZFC is obtained by expressing these axioms by sentences of $\mathbb{FOL}(\in)$, and it requires infinitely many sentences, because the Replacement Axiom **5** requires an *axiom scheme*. Here we will describe them briefly and informally, with a few interspersed comments.

1. *Extensionality.* Two sets are equal exactly when the have the same members.

2. *Empty set and Pairing.* There is a set $\emptyset$ with no members, and for any two sets a, b, there is a set $\{a, b\}$ whose members are exactly a and b.

3. *Unionset.* For each set A, there is a set $\cup A$ whose members are the members of the members of A,

$$t \in \cup A \Leftrightarrow (\exists x)[x \in A \,\&\, t \in x].$$

4. *Powerset.* For each set A, there is a set $\mathcal{P}(A)$ whose members are all the subsets of A.

An operation $F : \mathbb{V} \to \mathbb{V}$ is *definite* if it is *first order definable with parameters*, i.e.,

$$F(x) = G(x, a_1, \ldots, a_k)$$

where $G(x, y_1, \ldots, y_k)$ is first order definable, II-C.

5. *Replacement.* The image

$$F[A] = \{F(x) \mid x \in A\}$$

of a set A by a definite operation F is a set. (This was formulated in the 30s, primarily by Skolem, and it is much stronger than Zermelo's original *Separation Axiom.*)

For the next two axioms, we need the notion of *function* $f : A \to B$ from one set to another, which is not among our primitives, and so we need to "reduce" the notion of function to that of set. The trick is well known: first we fix some *ordered pair operation* (x, y) which satisfies the key property

$$(x, y) = (x', y') \Leftrightarrow x = x' \,\&\, y = y', \tag{18}$$

and then we model a function f by its graph,

$$G_f = \{(x, y) \in A \times B \mid y = f(x)\},$$

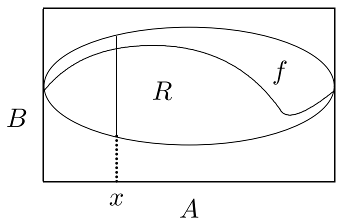
which is just a set with some special properties. It is common to use the so-called *Kuratowski pair operation*

$$(x, y) = \{x, \{x, y\}\},$$

but there are many others, and all that is needed is *some* operation which satisfies (18).

6. *Infinity.* There is a set I and a one-to-one function $f : I \to I$ which is not onto I, $f[I] \subsetneq I$.

Next comes Zermelo's chief contribution:



7. *Axiom of Choice* (AC). For each binary relation $R \subseteq A \times B$,

$$\begin{aligned} &(\forall x \in A)(\exists y \in B)(x, y) \in R \\ \Longrightarrow\ &(\exists f : A \to B)(\forall x \in A)(x, f(x)) \in R. \end{aligned}$$

In effect, AC postulates a function f which makes a choice $f(x)$ from the non-empty set $\{y \mid (x, y) \in R\}$, "simultaneously", for each $x \in A$. If B carries a well ordering $\leq$, we could take

$$f(x) = \text{the } \leq\text{-least } y \text{ such that } (x, y) \in R;$$

Zermelo showed that, conversely, AC *implies that every set is well orderable*, and identified numerous examples where the seemingly controversial AC is routinely used in mathematics. Somewhat later Hartogs showed that AC is also equivalent with the *cardinal comparability property*

$$(\forall A, B)[A \leq_c B \vee B \leq_c A]$$

without which there is no cardinal arithmetic, and this limited further opposition to AC to those who were willing to abandon completely Cantor's Paradise.

The last axiom of ZFC involves the *cumulative hierarchy of sets*, which is defined by recursion on the ordinal numbers as follows:

$$\begin{aligned} V_0 &= \emptyset, \\ V_{\alpha+1} &= \mathcal{P}(V_\alpha), \\ V_\lambda &= \textstyle\bigcup_{\alpha<\lambda} V_\alpha \quad (\text{if } \lambda \text{ is limit}). \end{aligned}$$

8. *Foundation.* Every set is a member of some V_α.

This is a limiting axiom, not needed for the development of Cantor's set theory or its applications, but it is important because it codifies within the axiomatic theory a conception of set which replaced in the 1930s Cantor's free-wheeling notion of a "collection into a whole": each set is reached starting with "nothing" (the emptyset $\emptyset$), by "indefinite" (never ending) "iteration" of the powerset operation. Admittedly more complex than Cantor's, this notion of *grounded set* prohibits the circular constructions which lead to the paradoxes, and it can be described intuitively in sufficiently clear terms to justify the axioms.

To see how classical mathematics can be developed on the basis of these seven axioms, consider first arithmetic. A *number system* is a triple $(\mathbb{N}, 0, S)$ such that $\mathbb{N}$ is a set, $0 \in \mathbb{N}$, $S : \mathbb{N} \to \mathbb{N}$ is a one-to-one function which is never 0, and

$$[X \subseteq \mathbb{N} \,\&\, 0 \in X \,\&\, S[X] \subseteq X] \implies X = \mathbb{N};$$

we prove that *there exists a number system* and that *every two number systems are isomorphic*, and then we choose some specific number system and call its members *the natural numbers*. The real numbers are identified with some *complete ordered field*, once we prove that one such exists and any two are order isomorphic, and so forth for other structures. This process of "defining" (more accurately: *modeling faithfully*) mathematical structures in set theory has found such widespread acceptance in mathematics, that "to make a notion precise" is now viewed as synonymous with "defining it in set theory".

D. **Independence Results**

It is, perhaps, ironic, that the axiomatization of set theory made possible to formulate and prove its own limitations. Let ZF be the theory with axioms **1**–**6**, i.e., without the Axiom of Choice:

Theorem. If ZF is consistent, then so are the theories ZFC+GCH (Gödel, 1938) and ZFC+¬CH (Paul Cohen, 1963).

In effect, ZFC can neither disprove nor prove the Continuum Hypothesis, unless a contradiction can be obtained from its "constructive" core. In addition, Cohen showed that ZF cannot prove the Axiom of Choice, and several additional consistency and independence results.

Gödel's proof uses an *inner model*, a subcollection of our intended universe of sets $\mathbb{V}$: using only axioms **1**–**6**, he defines a certain collection L of *constructible sets* and shows that if we re-interpret "set" to mean "member of L", then all the axioms of ZFC as well as GCH are true. Cohen's *forcing method* builds "virtual universes" which are "larger" than $\mathbb{V}$, and so he must describe them indirectly. This can be done with *Boolean-valued models*: a collection $M \subset \mathbb{V}$ and a binary condition E on M are defined, and then it is shown that, for a certain (complete) Boolean algebra B, the *Boolean semantics* of the "structure" (M, E) assign 1 to all the theorems of ZFC but something other than 1 to CH.

In both of these proofs, logic plays an essential role which goes much beyond providing the context in which their claims can be made precise. For example, the constructible universe L is defined by iterating the operation of taking all *first order definable* subsets rather than $\mathcal{P}(A)$ in the cumulative hierarchy of sets, and then a strong version of the Skolem-Löwenheim Theorem is used at a crucial point to show that GCH holds in L. Through the work, initially, of Robert Solovay for forcing and Ronald Jensen for constructibility, these theories have been much generalized and continue to be very active research areas of logic, with important applications to analysis, algebra and topology.

E. **Current Research in Set Theory**

In one direction, set theory is more involved now with applications than ever before. Especially fruitful has been the development in the 1960s of *effective descriptive set theory*, which incorporates methods from recursion theory into the study of definability on the continuum to yield very substantial applications to analysis.

Beyond the applications, set theory has attempted to confront the fundamental problem posed by the independence results: what does one do with the Continuum Problem, now that we know that it cannot be settled in ZFC? Some have adopted a formalist view, that it is meaningless to ask "whether CH is true or not", and that "set theory is the study of all models of ZFC". This is a very active area of research.

In another direction, people have looked for *new axioms*, extending ZFC, which might provide the needed answers, and a great deal of research has been done in this direction since the 1960s. Generally speaking, two kinds of axioms have been considered. *Large cardinal axioms* are plausible generalizations of the Axiom of Infinity, which, however, have very few direct consequences for the continuum. *Determinacy hypotheses* postulate that certain (fairly simple) infinite games on the natural numbers are *determined*; somewhat technical and not especially plausible, these axioms answer most definability questions about the real numbers that are independent of ZFC, although, unfortunately, they cannot settle the Continuum Problem. In a fundamental advance made in the 1980s, Donald A. Martin, John Steel and Hugh Woodin showed that the plausible large cardinal axioms imply the fruitful determinacy hypotheses, and so a "unified", very strong extension of ZFC has been created which is the subject of much current research. Unfortunately, it does not solve the Continuum Problem, and so the search goes on.

It may well be that set theory will continue to be dominated in the 21st century by the search for an answer to the Continuum Problem, as it certainly was during the century just ended.

References

Handbook of Logic in Computer Science, edited by S. Abramsky, T. S. E. Maibaum and D. M. Gabbay, in five volumes, the first publiahed in 1993, Clarendon Press, Oxford.

Handbook of Proof Theory, edited by Samuel R. Buss, Studies in Logic and the Foundations of Mathematics, vol. 137, Elsevier, 1998.

Model Theory, by Wilfried Hodges, Encuclopedia of Mathematics and its Applications, vol. 42, Cambridge University Press, 1993.

Theory of Recursive Functions and Effective Computability, by H. J. Rogers, Jr., McGraw-Hillm New York, 1967.

Set Theory, by Kenneth Kunen, Studies in Logic and the Foundations of Mathematics, vol. 102, Elsevier, 1998.

Descriptive Set Theory, by Yiannis N. Moschovakis, Studies in Logic and the Foundations of Mathematics, vol. 100, North Holland, 1980.

University of California, Los Angeles, and University of Athens