J. Austral. Math. Soc. 24 (Series A) (1977), 139–161

CONCISE SURVEY OF MATHEMATICAL LOGIC

JOHN STILLWELL

(Received 2 February 1976)
Communicated by J. N. Crossley

Abstract

This paper contains proofs of the compactness, completeness and Löwenheim–Skolem theorems for predicate logic, together with their application to non-standard numbers; proofs of undecidability in predicate logic and number theory, and the Gödel incompleteness theorem.

Subject classification (Amer. Math. Soc (MOS), 1970): 02—02

Introduction

The object of this survey is to present all the classic results of mathematical logic within the space of a single paper. In fact, a paper with this objective already exists (Kleene, 1958), and has influenced Sections 1 and 2 of the present work. However, Kleene's paper lacks adequate detail on non-standard numbers and the celebrated undecidability and incompleteness results, so we have attempted to fill this gap in the remaining sections of the present paper. Section 3, on non-standard numbers, draws on the exposition of Skolem (1955), while Sections 4 and 5, on undecidability, make some simplifications in the author's work previously published in Crossley and others (1972) and the work of Smullyan (1961).

Thanks to 30 years of technical simplifications (together with some shift in the meaning of "predicate logic" and "formal system") it is now possible to give short, complete proofs of undecidability for predicate logic and number theory (see Section 5). Unfortunately, these results have remained a mystery to the general mathematician because only the difficult methods originated by the pioneers (mainly Gödel) have received wide circulation. (To get the flavour of the original work the reader can consult Davis (1965), which contains all the classic papers in this field.) A much simpler approach is available through the work of Post (1944, 1947), but even today this has not been widely adopted. We hope that Section 5 will convince the reader of the fruitfulness of Post's Turing machine approach.

Gödel's most famous result, the incompleteness theorem, is the only proof where we omit some detail. The heart of the proof, the translation of a sentence about a machine M into a sentence ψ_M of number theory *is* done in complete detail, our omission is essentially the construction of a machine to write down the sentence ψ_M, this construction is neither difficult nor interesting.

The only mathematical prerequisites required for this paper are some familiarity with Cantor's methods of enumeration and diagonalization. Actually the only enumeration method we use is that for enumerating words over a fixed finite alphabet—first list the one-letter words, then the two-letter words, and so on. We appeal to this enumeration in order to claim that certain sets are countable when their members have received names in some fixed alphabet, in particular terms built from function symbols and constants. This is the key to the Löwenheim–Skolem theorem of Section 2.

We use the diagonal argument in Section 4 when we apply a Turing machine M to its own description $\ulcorner M \urcorner$, and it is the key to the undecidability results of Section 5. The reader will observe a contradiction arise in this "self-reference" situation exactly as it does in Cantor's proof that each set has a smaller cardinality than its power set.

1. Propositional logic

The logic of propositions or truth functional logic is concerned with analysing the truth of compound propositions in terms of the truth values of atomic propositions and the connectives which link them, such as "and", "or", "not".

We use letters $p_1, p_2, p_3, \ldots$ to denote atomic propositions. "True" and "False" are denoted by T and F.

A *truth function* or *connective* $f(p_1, \ldots, p_n)$ is simply any function: $\{T, F\}^n \to \{T, F\}$, and can be given a *truth table*

p_1	p_2	$\cdots$	p_n	$f(p_1 \cdots p_n)$
T	T		T	function
		$\vdots$		
F	F		F	values

of 2^n rows and $n+1$ columns, where the first n columns contain all possible values of $p_1, \ldots, p_n$, and the last column contains the function values.

"And", "or" and "not" are denoted by $\&, \vee, \neg$. They have the following truth tables

p_1	p_2	$p_1 \& p_2$
T	T	T
T	F	F
F	T	F
F	F	F

p_1	p_2	$p_1 \vee p_2$
T	T	T
T	F	T
F	T	T
F	F	F

p_1	$\neg p_1$
T	F
F	T

Formulae

Compound propositions can be represented by *formulae*, defined inductively as follows:

(a) any letter p_i alone is a formula (*atomic formula*);

(b) if φ and ψ are formulae so are $(\varphi \mathcal{E} \psi)$, $(\varphi \vee \psi)$, $\neg(\varphi)$, for example, $\neg(p_1)$, $(p_2 \vee \neg(p_1))$ and $(p_3 \mathcal{E} (p_2 \vee \neg(p_1)))$ are formulae.

Because of the bracketing, the component formulae φ and ψ in a formula constructed by (b) can be uniquely reconstructed. If φ and ψ are themselves compound, their components can likewise be reconstructed, until we get back to atomic formulae. Then if certain truth values have been assigned to the atomic formulae, the truth values of the successive compounds can be read off from the truth tables for $\mathcal{E}, \vee, \neg$, giving us in the end the truth values of the formula we started with. For example, evaluate $(p_3 \mathcal{E} (p_2 \vee \neg(p_1)))$ with $p_1 = F$, $p_2 = T$, $p_3 = T$:

$$\begin{array}{lcccccc}
 & T & & T & & & F \\
 & & & & & T & \\
 & & & & T & & \\
\text{The value is:} & & T & & & &
\end{array}$$

(This layout results from putting each truth value underneath the connective which determines it.)

Actually, some brackets are redundant, and we omit them where an unambiguous reading is still possible. For example,

$$\neg p_1 \mathcal{E} p_2 \quad \text{for} \quad (\neg(p_1) \mathcal{E} p_2)$$

and

$$p_1 \mathcal{E} p_2 \mathcal{E} p_3 \quad \text{for} \quad ((p_1 \mathcal{E} p_2) \mathcal{E} p_3)$$

and also

$$\text{for} \quad (p_1 \mathcal{E} (p_2 \mathcal{E} p_3))$$

since these are the same truth function.

All other truth functions may be compounded from $\mathcal{E}, \vee, \neg$. We leave this result as an exercise, with just the hint that an arbitrary truth table has a natural description in terms of $\mathcal{E}, \vee, \neg$.

An important example is $p_1 \Rightarrow p_2$, "if p_1 then p_2", which is the same truth function as $\neg p_1 \vee p_2$. Similarly for $p_1 \Leftrightarrow p_2$ ("if and only if"), which is $(p_1 \Rightarrow p_2) \mathcal{E} (p_2 \Rightarrow p_1)$.

Satisfaction and validity

An *interpretation* $\mathscr{I}$ of a formula $\varphi(p_1 \ldots p_n)$ is an assignment of values (T or F) to $p_1, \ldots, p_n$. $\mathscr{I}$ *satisfies* φ, written $\mathscr{I} \models \varphi$, if $\varphi = T$ under interpretation $\mathscr{I}$ (if $\varphi = F$ $\mathscr{I}$ falsifies φ), φ is *satisfiable* if some $\mathscr{I} \models \varphi$, φ is *valid* (or a *tautology*) if all $\mathscr{I} \models \varphi$. For example, $p_1 \vee \neg p_1$ is a tautology.

Since $\varphi(p_1, \ldots, p_n)$ has only 2^n interpretations, we can mechanically test for both satisfaction and validity, using truth tables. This method of establishing validity was given by Post (1921), where most of the basic problems of propositional logic are solved.

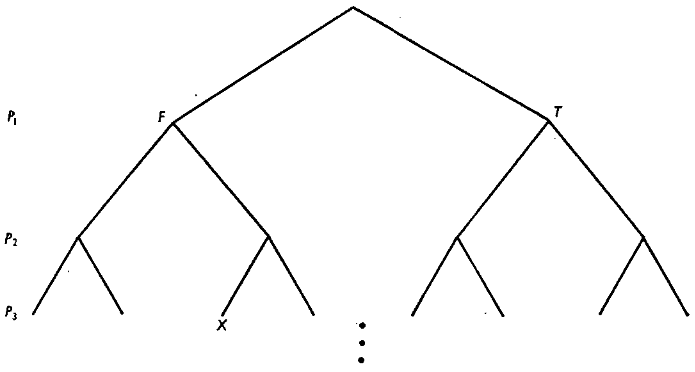
Note that satisfiability and validity are connected by: φ is valid if and only if $\neg\varphi$ is not satisfiable. It turns out that satisfiability is the better notion to work with.

Compactness theorem

Propositional logic is not adequate for mathematics in itself, since its formulae cannot express properties or relations between objects. What is surprising, however, is that mathematical sentences can be replaced by infinite *sets* of propositional formulae, then a reduction to finite sets can be made by the following theorem. (A set of formulae, Σ, is called satisfiable if there is an $\mathscr{I}$ which satisfies all formulae in Σ simultaneously.)

Compactness theorem. *If Σ is a set of (propositional) formulae, each finite subset of which is satisfiable, then Σ itself is satisfiable.*

Proof. All possible assignments to $p_1, p_2, p_3, \ldots$ can be represented on the tree shown. Each node at level n represents one of the 2^n possible assignments to $p_1, \ldots, p_n$. For example, the node marked x represents $p_1 = F$, $p_2 = T$, $p_3 = F$.



Now if the nodes are examined from the top down, and branches cut off at each node where the corresponding assignment falsifies some formula in Σ, the final result will be:

either (i) all branches terminate by some level n,

or (ii) there is an infinite branch.

An infinite branch gives a value to each p_n, and these values must satisfy each formula in Σ, otherwise the branch would be cut, so (ii) means Σ is satisfiable. (i) means that for each of the 2^n assignments to $p_1, \ldots, p_n$, some formula in Σ comes out F. Let these formulae be $\varphi^{(1)}, \varphi^{(2)}, \ldots, \varphi^{(k)}$ (so $k \leqslant 2^n$). But then *no* assignment makes $\varphi^{(1)}, \ldots, \varphi^{(k)}$ simultaneously T, that is, this finite subset of Σ is not satisfiable.

EXERCISE. *Redefine "interpretation" to be an infinite sequence of* 0'*s and* 1'*s, that is, a point in the space* 2^ω. *Show that each basic open set* B *in* 2^ω *(product topology) corresponds to a formula* φ, *namely* $B = \{\mathscr{I} \mid \mathscr{I} \models \varphi\}$ *and that the compactness of* 2^ω *follows from the compactness theorem.*

We defer until the next section the general procedure for reducing mathematical sentences to sets of propositional formulae. However, we can present an example which illustrates this fact, and affords an interesting application of the compactness theorem.

THEOREM. *If every finite map is* 4-*colourable then so is every infinite map.*

Let countries be denoted by $c_1, c_2, c_3, \ldots$. A map is determined by a set of atomic formulae $B(c_i, c_j)$, which denote that c_i borders on c_j. Other atomic formulae $K_n(c_i)$, $n = 1, 2, 3, 4$, denote that c_i has colour n. Then "colouring" the map means to satisfy the following formulae:

$K_1(c_i) \vee K_2(c_i) \vee K_3(c_i) \vee K_4(c_i)$ "each country has a colour"

$K_1(c_i) \Rightarrow (\neg K_2(c_i) \,\&\, \neg K_3(c_i) \,\&\, \neg K_4(c_i))$

$\vdots$ "and only one colour"

$K_4(c_i) \Rightarrow (\neg K_1(c_i) \,\&\, \neg K_2(c_i) \,\&\, \neg K_3(c_i))$

$B(c_i, c_j) \Rightarrow ((K_1(c_i) \Rightarrow \neg K_1(c_j)) \,\&\, \ldots \,\&\, (K_4(c_i) \Rightarrow \neg K_4(c_j)))$

"bordering countries have different colours"

for $i, j = 1, 2, 3, \ldots$ together with the border relations of the given map.

And these *can* be satisfied, because each finite subset mentions only finitely many countries and hence is satisfiable under the assumption that finite maps are 4-colourable.

EXERCISE. *Prove this theorem directly by a tree argument.* (Hint: Each node of the tree branches 4 ways.)

2. Predicate logic

To obtain a language suitable for the expression of mathematical theories we need the following elements:

Variables $x_1, x_2, x_3 \ldots$	intended to range over the domain of the given theory
Predicate letters: $P_1, P_2 \ldots$	to denote relationships on the domain
Function letters: $f_1, f_2 \ldots$	for functions on the domain
Constants: $a_1, a_2, \ldots$	for distinguished objects
Quantifiers: $\forall x_i$, $\exists x_i$	"for all x_i", "there exists an x_i"

together with $\&, \vee, \neg$ and also $=$, with their usual interpretation.

The functions, constants and $=$ are included for their convenience and in principle can be eliminated, since functions and $=$ are particular types of relations, and a constant a_i can be replaced by a relation $A_i(x)$ which denotes $x = a_i$. However, we do not carry this out, since functions in particular will be very useful.

Formulae and interpretations

Formulae are constructed inductively, first we need the notion of a *term*:

(a) a variable or a constant is a term;

(b) if $t_1, \ldots, t_n$ are terms, then so is $f_i(t_1, \ldots, t_n)$.

Now, an *atomic formula* is simply $P_i(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms. And if φ and ψ are formulae so are

$$(\varphi \mathbin{\&} \psi), \quad (\varphi \vee \psi), \quad \neg(\varphi), \quad \forall x_i(\varphi), \quad \exists x_i(\varphi).$$

(If we want $\forall x_i \varphi$ or $\exists x_i \varphi$ to mean something we will want x_i to be *free* in φ, that is, not governed by any other quantifier.)

An *interpretation* $\mathscr{I}$ is a sequence $(D, \bar{P}_1, \bar{P}_2, \ldots, \bar{f}_1, \bar{f}_2, \ldots, \bar{a}_1, \bar{a}_2, \ldots)$ where D is the domain and the $\bar{P}_i, \bar{f}_i, \bar{a}_i$ are relations and functions on D, and objects in D, corresponding to the predicate, function and constant symbols.

$\mathscr{I}$ *satisfies* a formula φ, $\mathscr{I} \models \varphi$, if φ is T under $\mathscr{I}$ when $\forall x_i$, $\exists x_i$ respectively mean "for all members of D" and "there exists a member of D".

φ is *valid* if T under all interpretations. Examples of valid formulae are $\forall x_1 P(x_1) \Rightarrow \exists x_1 P(x_1)$ and $\exists x_1 \forall x_2 P(x_1, x_2) \Rightarrow \forall x_2 \exists x_1 P(x_1, x_2)$. (Note that there is no obvious test for validity in this language, since each formula has infinitely many interpretations. Later we shall show that in fact there is no algorithm for deciding validity in predicate logic.)

Languages for certain mathematical theories

(i) *Group theory*: Requires one constant, e (for the identity element) and a two-place function $\cdot$ (for group multiplication).

Exercise. *Write down the group axioms in this language. A* group *is in fact defined to be any interpretation which satisfies these axioms.*

(ii) *Number theory*: Requires a constant 0, one-place function $'$ (successor) and two-place functions $+$ and $\cdot$. The terms $0, 0', 0'', \ldots$ then denote the natural numbers.

Exercise. *Define the following relations in this language*:

$$x < y, \quad x \textit{ divides } y, \quad x \textit{ is prime}.$$

(It is now known, essentially from the work of Gödel (1931), that all computable functions are definable in this language.)

(iii) *Set theory* (for example, Zermelo–Fraenkel). It requires only a single binary relation $\in$ (membership), and all contemporary mathematics can be developed within this framework.

Prenex and Skolem form

It is now clear that mathematical sentences can be viewed as formulae in the language of predicate logic. So the problem of reducing them to propositional formulae amounts to finding a procedure for reducing predicate formulae.

The first step in this procedure is to move all quantifiers to the front (giving the so-called *prenex form*). This can be done by systematic application of the following equivalences:

$$\neg\forall x\varphi \Leftrightarrow \exists x\neg\varphi$$
$$\neg\exists x\varphi \Leftrightarrow \forall x\neg\varphi$$
$$\varphi \,\&\, \forall x\psi \Leftrightarrow \forall x(\varphi \,\&\, \psi)$$
$$\varphi \,\&\, \exists x\psi \Leftrightarrow \exists x(\varphi \,\&\, \psi)$$

(and similarly with $\vee$ in place of $\&$). The rules for moving the quantifier in front of $\&$, $\vee$ require that x is not free in φ, so if necessary x should be changed to a new variable not free in φ before moving the quantifier forward.

The next step is to remove the $\exists$ quantifiers in favour of constant and function symbols. At this stage we seek, not a logical equivalent, but a formula which is satisfiable if and only if the original formula is satisfiable. The simplest cases are the following:

$\exists x\varphi(x)$ satisfiable if and only if $\varphi(a)$ holds for some object a,
if and only if $\varphi(a_i)$ is satisfiable;
$\forall x\exists y\varphi(x,y)$ satisfiable if and only if $\forall x\varphi(x,f(x))$ holds for some function f,
if and only if $\forall x\varphi(x,f_i(x))$ is satisfiable.

The general prenex form can be handled by simple extension of this idea, for example

$\forall x\exists y\forall z\exists tP(x,y,z,t)$ satisfiable if and only if
$\forall x\forall z\exists tP(x,f(x),z,t)$ satisfiable
if and only if
$\forall x\forall zP(x,f(x),z,g(x,z))$ satisfiable.

The result of applying these reductions to a formula φ is called φ^S, the *Skolem form* of φ, and the functions appearing in φ^S are called *Skolem functions*.

Example. $\forall x\exists yR(x,y)$ is satisfiable in $\langle \mathbf{R}, < \rangle$ where $\mathbf{R} = \{\text{real numbers}\}$. Its Skolem form is $\forall xR(x,f(x))$ and a suitable Skolem function is $f(x) = x+1$.

Löwenheim–Skolem and Herbrand theorem

We have that φ is satisfiable if and only if φ^S is satisfiable. Now φ^S is a *universal* formula, that is, of the form $\forall x_1 \ldots \forall x_n\varphi(x_1,\ldots,x_n)$ where φ is quantifier-free. To say $\forall x_1 \ldots \forall x_n\varphi(x_1,\ldots,x_n)$ holds in some D is simply to say that each *instance*

$\varphi(d_1,\dots,d_n)$ with $d_1,\dots,d_n \in D$ is true, so automatically $\forall x_1 \dots \forall x_n \varphi(x_1,\dots,x_n)$ will hold in any $D' \subseteq D$. The only proviso is that D' must be closed under any functions occurring in φ, so that whenever the variables denote members of D', the terms in φ do also.

This means that if φ^S holds in D we can find a $D' \subseteq D$ in which φ^S holds simply by taking any $d \in D$ and generating all the *Skolem terms* compounded from d and the Skolem functions. Note that this interpretation will satisfy φ as well.

To sum up: φ is satisfiable

if and only if φ^S is satisfiable;

if and only if φ^S is satisfiable in the domain of Skolem terms;

if and only if the instances of φ^S in the domain of Skolem terms are simultaneously satisfiable.

Two important theorems follow from these facts.

LÖWENHEIM–SKOLEM THEOREM. *If φ is satisfiable then it is satisfiable in a countable domain* (Löwenheim, 1915).

PROOF. We have φ is satisfiable if and only if φ is satisfiable in the domain of Skolem terms. But the Skolem terms are words on a finite alphabet (namely d, the Skolem functions of φ and brackets), hence they form a countable set.

EXAMPLE. We have $\forall x \exists y R(x,y)$ holds in $\langle \mathbf{R}, < \rangle$, and a Skolem function $f(x) = x+1$. Choosing $0 \in \mathbf{R}$, the domain of Skolem terms is $\{0,1,2,\dots\}$, hence countable, and $\forall x \exists y R(x,y)$ holds in this domain.

EXERCISE. $\forall x \forall y \exists z (R(x,y) \Rightarrow (R(x,z) \mathcal{E} R(z,y)))$ *also holds in* $\langle \mathbf{R}, < \rangle$. *Find a suitable Skolem function and describe the domain of Skolem terms.*

HERBRAND'S THEOREM. *φ is valid if and only if some finite set of instances of $(\neg\varphi)^S$ is unsatisfiable* (Herbrand, 1930).

PROOF. φ is valid

if and only if $\neg\varphi$ is unsatisfiable;

if and only if the set of instances of $(\neg\varphi)^S$ is unsatisfiable;

if and only if some finite set of instances is unsatisfiable

(since the instances are propositional formulae, and so compactness of propositional logic applies).

This theorem gives a procedure for reducing a predicate formula φ to a propositional formula, admittedly it works only when φ is valid. Namely, construct $\neg\varphi^S$ and enumerate the Skolem terms and instances of $\neg\varphi^S$. Test each finite conjunction of these instances for satisfiability. Termination will occur whenever φ is valid, and can be regarded as a "proof" of φ.

This is the essence of the so-called *completeness* theorem of Gödel, which says that there is a consistent formal system in which every valid formula of predicate logic has a proof (Gödel, 1930).

For our purposes, "consistent formal system" can be regarded as a mechanical procedure which generates only valid formulae. The above procedure can obviously be made sufficiently precise to fit this definition.

REMARKS. The Löwenheim–Skolem theorem (1915) was the first result to reveal an inadequacy in the axiomatic method. Namely, axioms do not always completely determine their intended subject matter. In particular, no axiom φ can completely characterize an uncountable set, since φ is also satisfiable in a countable domain.

On the other hand, it is the fact that we really only have to look at countable models that makes the completeness theorem possible, for the compactness theorem gives a reduction from the countable to the finite, and thus allows the search for a proof to terminate.

3. Compactness and non-standard numbers

Our results so far have depended on only two ideas—compactness for propositional logic and the Skolem form of a predicate formula. In this section we show how a number of results may be established on the same basis.

These results are part of *model theory*, so called because it studies the connections between formulas and their *models* (the interpretations which satisfy them). The Löwenheim–Skolem theorem (1915) was the first result of model theory to be discovered. We can now give its generalized form, due to Skolem.

GENERALIZED LÖWENHEIM–SKOLEM THEOREM. *If* Σ *is a satisfiable set of formulas, then* Σ *has a countable model.*

PROOF. It is easily seen that the language of predicate logic can be based on a finite alphabet (using the subscript numerals as separate letters), therefore any set of predicate formulas φ is countable.

Then the associated set of Skolem functions is also countable, and likewise the set D' of Skolem terms generated by these Skolem functions.

But then each $\varphi \in \Sigma$ holds in D' because D' is closed under the Skolem functions for φ, hence D' is a countable domain in which all members of Σ hold simultaneously.

NOTE. This theorem means that even mathematical theories with infinitely many axioms (such as Zermelo–Fraenkel set theory) have countable models. This seems paradoxical, since the sentence "There exists an uncountable set" can be proved in set theory. The explanation is that when "S is uncountable" holds in a model it merely means that there is no function *in the model* which maps S one–one onto the natural numbers.

COMPACTNESS THEOREM. *If Σ is a set of (predicate) formulas, each finite subset of which has a model then Σ itself has a model.*

PROOF. Convert each φ in Σ to φ^S and enumerate its instances $\varphi_1^S, \varphi_2^S, \ldots$ in the domain of Skolem terms. Since the instances are propositional formulas, we can get the theorem from compactness of propositional logic, provided we are able to show that each finite subset of $\{\varphi_i^S\}_{\varphi \in \Sigma, i=1,2,3,\ldots}$ is satisfiable.

But each finite subset mentions only finitely many φ's from Σ, so in fact any number of instances of their Skolem forms are simultaneously satisfiable, by assumption that finite subsets of Σ are satisfiable.

NON-STANDARD INTEGERS

We can see from the above proof that predicate logic compactness depends not only on propositional logic compactness but also on Skolem functions. Thanks to the latter ingredient, some rather difficult results become quite easy consequences of predicate logic compactness. A good example of this is the construction of non-standard numbers.

Let Σ be any set of true formulas in the language of number theory, which for convenience we will take to include the $<$ relation, and numerals $0, 1, 2, \ldots$. Let c be a new constant, and add to Σ the following infinite set of formulas:

$$\begin{gathered} c > 0 \\ c > 1 \\ c > 2 \\ \vdots \end{gathered}$$

The enlarged set, Σ', has the property that any finite subset is satisfiable, because there will be a bound N to the n mentioned in those sentences "$c > n$" which occur, so the given finite subset can be satisfied by taking $c = N+1$.

Then by compactness, Σ' as a whole is satisfiable, and the model must include an object $\bar{c}$ with infinitely many preceding objects $\bar{0}, \bar{1}, \bar{2}, \ldots$. This model is called a *non-standard model* for number theory, and $\bar{c}$ is called a non-standard number.

SKOLEM'S NON-STANDARD MODEL

The indirectness of the above argument makes it next to impossible to visualize what the non-standard model looks like. Skolem gave an interesting direct construction when he first proved this result in 1934, which at the same time clarifies the rôle of Skolem functions in the proof.

Skolem for technical reasons considers the theory of positive integers, so that 0 is dropped. He converts each φ in Σ to φ^S, and also eliminates $\mathcal{E}, \vee, \neg$ from the quantifier-free part, so that it becomes a pure *equation*. This is possible because of the following equivalences:

$$\neg a = b \Leftrightarrow (\exists x)(a+x = b \vee a = b+x)$$

(this is why we do not want 0 in the system),

$$a = b \vee c = d \Leftrightarrow ac+bd = bc+ad,$$
$$a = b \,\&\, c = d \Leftrightarrow a^2+b^2+c^2+d^2 = 2ab+2cd$$

(the latter equivalences come from multiplying out

$$(a-b)(c-d) = 0 \quad \text{and} \quad (a-b)^2+(c-d)^2 = 0$$

respectively, and rearranging so as to get all positive terms).

The Skolem functions, incidentally, can be explicitly defined since the y corresponding to the x in a $\forall x\ \exists y$ situation can be taken as the *least* such y. (In case Σ consists of *all* true sentences of the theory of $+$ and $\times$, the Skolem functions are the so-called *arithmetically definable* functions.)

Now with the axioms in the form of equations, it is clear that they are satisfied not only by numbers, but also by functions, since if, say, $F(n) = G(n)$ for all n then also $F(f(m)) = G(f(m))$ for any function value $f(m)$, and this is what it means for $F(f) = G(f)$ to hold.

We might jump to the conclusion that the set of functions constitute the non-standard model, with the constant functions $f = 1$, $f = 2$, ... playing the rôle of the standard numbers, and other functions, for example $f(n) = n$, being the non-standard elements. However, we note that the equivalence

$$\neg a = b \Leftrightarrow (\exists x)(a+x = b \vee a = b+x)$$

is *not* true for functions, so we cannot carry the functions model back to our original set of sentences Σ.

The difficulty is that functions are not *comparable*—we cannot in general say that one function is less than another. Skolem's solution to this difficulty, which takes the place of a compactness argument, is to cut down the domain to a certain infinite subset of the positive integers.

To begin with, we need a countable set $f_1, f_2, \ldots$ of functions to satisfy the equations. In fact the set of one-place arithmetically definable functions is such a set, but we could start with any function and close under all the Skolem functions derived from Σ.

Now since only three relations $>$, $<$, $=$ can hold between $f_1(n)$ and $f_2(n)$, at least one of them must hold infinitely often. Take the first which does so, and let D_1 be the set of n for which it holds. f_1 and f_2 are then *comparable* on the set D_1.

To get comparability of $f_1, f_2, \ldots, f_{k+1}$ we assume we have an infinite set D_{k-1} on which $f_1, f_2, \ldots, f_k$ are comparable. Then there are only finitely many possible order relations between $f_{k+1}(n)$ and $f_1(n), \ldots, f_k(n)$, so one of them must occur for infinitely many n in D_{k-1}. Take the first which does so, and let D_k be the set of n in D_{k-1} for which it holds. Thus we have infinite sets

$$D_1 \supseteq D_2 \supseteq D_3 \supseteq \dots$$

such that $f_1, f_2, \dots, f_{k+1}$ are comparable on D_k, for all k.

Now consider an infinite set D_∞ obtained as a sequence

$$\begin{aligned} d_1 &= \text{least member of } D_1, \\ d_{k+1} &= \text{least member of } D_{k+1} > d_1, d_2, \dots, d_k. \end{aligned}$$

Then any f_i, f_j are comparable *at all but a finite number* of points in D_∞. (In fact, for all d_k with $k \geqslant \max(i,j)$ because all these d_k are in $D_{\max(i,j)}$, where f_i, f_j are comparable.)

Now if we take equivalence classes

$$[f_i] = \{f \mid f = f_i \text{ at all but finitely many points of } D_\infty\},$$

we have finally comparable objects, since

$$\begin{aligned} &[f_i] < [f_j] \\ &\text{or } > \\ &\text{or } = \end{aligned}$$

$$\text{means} \quad \begin{matrix} f_i(n) < f_j(n) \\ > \\ = \end{matrix} \quad \text{at all but finitely many points of } D,$$

and we know that exactly one of these cases always holds.

These objects $[f_i]$ are the elements of Skolem's model. [1], [2], [3], ... correspond to the standard positive integers, while, for example, $[f]$ for $f(n) = n$ is a non-standard integer, since $[f] > [1]$, $[f] > [2]$, $[f] > (3]$, ...

EXERCISE. *Borel* (1952) *postulates a property* $Ac(x)$, "*x is accessible*" *satisfying:* $Ac(1)$, $Ac(x) \Rightarrow Ac(f_i(x))$ *for each* "*definable*" f_i, *and* $\exists x \neg Ac(x)$. *Show that the non-standard elements of Skolem's model can be viewed as the* "*inaccessible*" *numbers in this sense, if* "*definable*" *is interpreted as* "*arithmetically definable*".

INFINITESIMALS

If we base a theory of rational and real numbers on number theory then this theory will also have non-standard models. The reciprocal of a non-standard integer $> 1, 2, 3, \dots$ will be an "infinitesimal" $< \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots$.

These infinitesimals behave much the same as Newton and Leibniz wanted them to, and thus many classical calculus arguments can be given a rigorous basis (although lacking the innocent simplicity of the originals).

EXERCISE. *If $\varepsilon_1, \varepsilon_2$ are infinitesimals, show that $\varepsilon_1+\varepsilon_2$, and $\varepsilon_1\alpha$, for any $\alpha<a$ standard number, are also infinitesimal.*

Definitions of limit and continuity take an especially appealing form, because it turns out that "$f(x)\to l$ as $x\to a$" means that $|x-a|$ is infinitesimal $\Rightarrow |f(x)-l|$ is infinitesimal. Using the result of the above exercise one can then prove the well-known theorems on sum and product of limits.

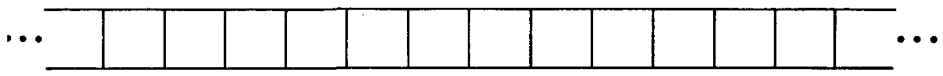
REMARKS. Skolem's model tells us roughly that if the standard integers correspond to constant sequences then non-standard integers are sequences which $\to\infty$. Infinitesimals therefore are sequences which $\to 0$.

This makes it rather easy to see how they work, since we already accept that real numbers are sequences (Cauchy sequences). Infinitesimals are simply null sequences, and the facts in the exercise correspond to the well-known results about sums and constant multiples of null sequences.

4. Turing machines

Around 1936 several attempts were made to capture the notion of "algorithm" or "computation" by a precise definition. It turned out eventually that all the proposed definitions were equivalent, and no algorithm known to the present day fails to satisfy them, so that we now accept that the word "algorithm" has a precise meaning.

Perhaps the most immediately convincing of the definitions proposed was Turing's concept of a *computing machine* (Turing, 1936). The machines work on an infinite tape divided into squares:



Each square can bear one symbol from a finite alphabet $\{\square, S_1, \ldots, S_m\}$, which depends on the machine given, and $\square$ denotes the blank square.

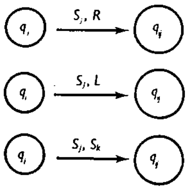
Only finitely many squares are initially marked, and the machine has a read/write head which can scan one square at a time and whose starting position is the leftmost marked square. Connected to the head is a control section which can assume finitely many *internal states* $q_1, \ldots, q_n$ (starting always in state q_1). The machine performs a discrete sequence of atomic acts, each uniquely determined by the current $\langle q_i, S_j\rangle$ (state, scanned symbol) pair, and of the form either

(i) move one square to right, and enter new state q_{ij} or

(ii) move one square to left, and enter new state q_{ij} or

(iii) replace S_j by S_k and enter new state q_{ij}.

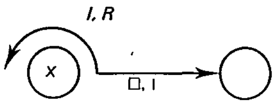
Turing machines can be conveniently represented by directed graphs with components



for the three types of act just described. The fact that each $\langle q_i, S_j\rangle$ determines at most one act means there is at most one arrow out of each node for each S_j in the alphabet. If there is no S_j arrow out of q_i this means that the machine halts when it reaches the $\langle q_i, S_j\rangle$ situation.

Notice that once the graph is completed, the q_i labels become redundant, since a state is determined by its position in the graph. Therefore we need actually mark the initial state only (say, with an X).

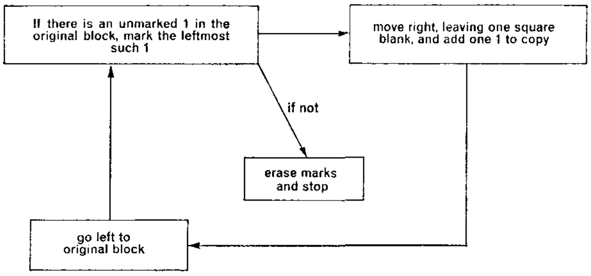
Example 1. Machine which adds a 1 at the right-hand end of a block of 1's.

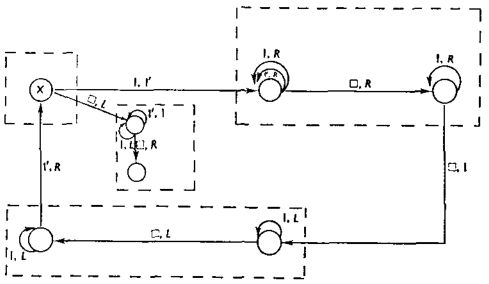


(This machine will work no matter where it is started on the given block; however, we make the convention of starting on the leftmost marked square, since in general a specific starting point is necessary.)

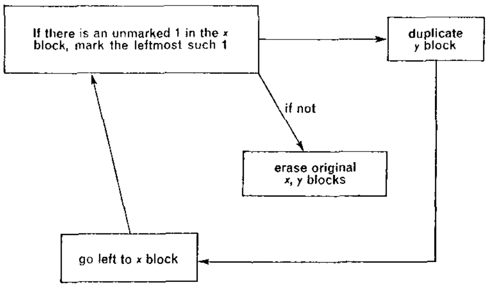
Example 2. Machine which duplicates a block of 1's.

The method is to mark each successive 1 in the original block (replace 1 by 1′) adding a 1 to the copy block as each mark is made. We first sketch a flow chart of the computation, then fill in the details of each box in the chart. (To follow these details, it is important to remember that the starting point is the leftmost square of the original block, hence the marked segment of the original is always on the left.)


If there is an unmarked 1 in the original block, mark the leftmost such 1
move right, leaving one square blank, and add one 1 to copy
if not
erase marks and stop
go left to original block



Example 2 is virtually as general as one needs to consider in constructing Turing machines, since it contains a loop which allows an operation ("add 1" in this case) to be iterated. To carry out multiplication we iterate duplication as follows: given blocks of length x, y produce a block of length xy by duplicating the y block x times, putting markers on the x block to count the correct number of duplications. The flow chart would be


If there is an unmarked 1 in the x block, mark the leftmost such 1
duplicate y block
if not
erase original x, y blocks
go left to x block

and the machine can easily be completed along the lines of Example 2.

Continuing in this vein it is easy to see how to compute many functions of positive integers, for example we get a block length x^y by iterating multiplication.

We call a function $f(x_1, \ldots, x_n)$ *computable* if there is a Turing machine M_f which, when started on a tape with blocks of $x_1, x_2, \ldots, x_n$ ones respectively (separated by single blank squares), eventually halts with $f(x_1, \ldots, x_n)$ ones on the tape. Examples 1 and 2 verify that $f(x) = x+1$ and $f(x) = 2x$ are computable functions.

Exercise. *Construct machines which compute* $2^x, x!$.

Solution of problems by algorithm

By a "problem" P we mean one consisting of an infinite class of questions Q; a solution is then a systematic, mechanical method or *algorithm* for correctly answering each question in the class P. For an algorithm to be applicable to the Q, each Q must be a string of symbols, or "word", in some fixed finite alphabet. If Q has an "intrinsic" meaning apart from its symbolic form as a word, then the word must be a "reasonable equivalent" of Q, and not contain additional information (such as the answer to Q!).

We then define P to be *solvable* if there is a Turing machine M, which, when started on any $Q \in P$, will eventually halt on 1 if the answer to Q is YES, and on $\square$ if the answer to Q is NO.

(Obviously this method of signalling YES and NO by Turing machine is quite arbitrary, but any other reasonable signals for YES and NO will be convertible by Turing machines into the ones we have chosen.)

Just as Turing machines can compute familiar functions, so they can implement familiar algorithms, for example the Euclidean algorithm for testing whether two numbers are relatively prime. However, our objective is to find an *unsolvable* problem.

Standard description

Another way to describe Turing machines is by a list of *quadruples*, of the three forms $q_i\, S_j\, Rq_{ij}$, $q_i\, S_j\, Lq_{ij}$, $q_i\, S_j S_k\, q_{ij}$ to denote the three types of atomic act (where there will be at most one quadruple for each pair $\langle q_i, S_j\rangle$). For example, the machine in Example 1 has two states, which we can call q_1, q_2, and its quadruples are then $q_1\ 1\ R\ q_1$, $q_1\ \square\ 1\ q_2$.

Without loss of generality we can assume that all Turing machine alphabet symbols are chosen from $\square$, 1, $1'$, $1''$, ... If we use q, q', q'', ... to denote states then any quadruple is a word on the alphabet $\{\square, 1, ', q, R, L\}$. So also is a machine, for we can simply write the quadruples of a given machine M end to end and still have an unambiguous description of M. If, finally, we code into an agreed machine alphabet as follows:

$$\begin{aligned} \square &\leftrightarrow \square \\ 1 &\leftrightarrow 1 \\ ' &\leftrightarrow 1' \\ q &\leftrightarrow 1'' \\ R &\leftrightarrow 1''' \\ L &\leftrightarrow 1'''' \end{aligned}$$

we get a word $\ulcorner M \urcorner$ which we call the *standard description* of M.

The beauty of the standard description is that it can be presented to the machine M itself, making a diagonal or self-referential construction possible.

The halting problem

Consider the following class of questions.

Q_M: Does M applied to $\ulcorner M \urcorner$ eventually halt on $\square$?

Suppose there is a machine S which solves this problem. It is reasonable to assume that S receives the question Q_M in the form of the word $\ulcorner M \urcorner$, since this contains all the necessary information. We quickly see that S then cannot correctly answer Q_S, for

(i) if S applied to $\ulcorner S \urcorner$ eventually stops on $\square$, then Q_S has answer YES, so S must stop on 1.

(ii) if S applied to $\ulcorner S \urcorner$ never stops on $\square$, then Q_S has answer NO, so S must stop on $\square$.

Thus there *is* no machine S to solve this problem. We have in fact shown that for any machine S which *partially* solves the problem—in the sense that S does not always give answers, but all its answers are correct—then Q_S is a specific question that S fails to answer.

There is in fact a machine which will correctly answer all the questions whose answer is YES. This is because there is a *universal* machine U which can simulate any machine M, thus if M eventually halts U can see whether it has halted on $\square$ or not. The unsolvable part of the problem is in tracking down the machines M which do not halt. (The universal machine and the halting problem were first discussed in Turing (1936), though not in exactly the above form.)

5. Undecidability and incompleteness

The fundamental problem for any mathematical theory is to decide whether a given sentence of the theory is true or false, this problem is called the *decision problem*. If no algorithm exists for the solution of this problem the theory is called *undecidable*. It turns out that most interesting mathematical theories are undecidable because the halting problem can be translated into them. We shall illustrate this phenomenon in the case of predicate logic and number theory, by giving specific translations of the questions Q_M into sentences of logic and number theory respectively.

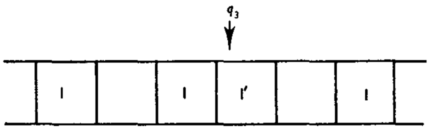
If the decision problem is unsolvable, we can still hope that a "semi-decision procedure" exists which will give the answer YES for each true sentence of the theory. In this case the theory is called *complete*. We have shown that predicate logic is indeed complete by giving an algorithm which terminates for each valid sentence. Traditional formal axiomatic systems can be viewed as attempts to find a semi-decision procedure. An axiom system is called complete if all true sentences can be deduced from the axioms by prescribed rules of inference. In this event, a systematic search for a given sentence φ among the consequences of the axioms will terminate just in case φ is true, so we have a semi-decision procedure.

Conversely, if a theory has no semi-decision procedure, that is, is incomplete in the algorithmic sense, then any axiom system for it must be incomplete. We will demonstrate incompleteness in the case of number theory (as a corollary of undecidability) by showing that no Turing machine can correctly give output YES for each true sentence of number theory.

Notice that incompleteness of (say) number theory means that for each axiom system there is a true sentence of number theory which is not provable in the system. This is called an *undecidable sentence* (relative to the given system). Unlike undecidability of the theory, undecidability of a single sentence is not absolute, since we may find methods outside the system to establish that the sentence is true. This will in fact happen in our number theory example; the point of incompleteness is that there is no end to the new methods that must be invented.

Representation of Turing machines by word transformations

In order to transform the "M on $\ulcorner M \urcorner$ halting" problem into one easily translated into logic and arithmetic we first construct a new representation of Turing machines. The computation situation at a given instant is described by a word ("situation word") which includes the marked portion of tape, the scanned square and the current state q_i. For example, we can describe the situation



by $* \, 1 \, \square \, 1 \, q_3 \, 1' \, \square \, 1 \, *$

where q_3 is inserted to the left of the scanned symbol, and $*$'s mark the ends of the used portion of tape.

Successive atomic acts of the machine will then correspond to changes in the situation word in the neighbourhood of the q symbol. It is easy to see that the following transformations bring about the correct sequence of changes:

Type of quadruple	Transformations
$q_i \, S_j \, S_k \, q_{ij}$	$q_i \, S_j \to q_{ij} \, S_k$
$q_i \, S_j \, R q_{ij}$	$q_i \, S_j \, S_k \to S_j \, q_{ij} \, S_k$ (for each S_k) $q_i \, S_j * \to S_j \, q_{ij} \, \square \, *$
$q_i \, S_j \, L q_{ij}$	$S_k \, q_i \, S_j \to q_{ij} \, S_k \, S^j$ (for each S_k) $* \, q_i \, S_j \to * \, q_{ij} \, \square \, S_j$

Notice that the q symbol always has a machine alphabet symbol on its right, and the $*$ in effect creates new blank squares when the q symbol approaches either end of the word.

A word representing a halt-on-$\square$ situation is one containing $q_h\square$, where $q_h\square$ is not in the left-hand side of any transformation. To simplify such words we introduce a new symbol $\diamondsuit$, which is created with the appearance of $q_h\square$ in the word and then proceeds to eat up all other symbols. Namely, we add the transformations

$$q_h \square \to \diamond$$

$$\left.\begin{array}{r} \diamond S \to \diamond \\ S \diamond \to \diamond \end{array}\right\} \text{ for each symbol } S.$$

These transformations (including those specific for the machine M) we call the *M-calculus*, and we write $W_1 \xrightarrow{M} W_2$ if W_1 is convertible to W_2 by a sequence of transformations in the M-calculus.

We now see that the question Q_M (Does M applied to $\ulcorner M \urcorner$ eventually halt on $\square$?) has the equivalent form

$$Is * \ulcorner M \urcorner * \xrightarrow{M} \diamond?$$

and therefore the unsolvability of the Q_M problem means there is no algorithm for deciding $* \ulcorner M \urcorner * \xrightarrow{M} \diamond$.

The latter formulation of the problem is particularly suitable for translation into both logic and number theory, since words can be represented by *terms* in predicate logic and by *numerals* in number theory. We shall sketch the translation into logic, then do the more difficult translation into number theory in detail.

Undecidability of predicate logic

Using $q, q', q'', \ldots$ for states and $\square, 1, 1', 1'', \ldots$ for machine symbols, all machine calculi can be based on words from the 6-letter alphabet $\{\square, 1, ', q, *, \diamond\}$. We take these letters as constants and form terms by a two-place function $f(x, y)$ written (xy) for simplicity. Terms are then just words with brackets in them, and we can effectively nullify the brackets by an associative axion

$$((xy)z) = (x(yz)). \qquad \ldots (1)$$

We can regard $\xrightarrow{M}$ as a two-place predicate symbol. The transformations of the M-calculus, $T \xrightarrow{M} T^*$, are implemented via axioms which allow substitution of T^* for T wherever it occurs in a word, namely

$$\forall x \, \forall y \, (xT \xrightarrow{M} xT^* \, \mathcal{E} \, Ty \xrightarrow{M} T^* y \, \mathcal{E} \, xTy \xrightarrow{M} xT^* y).$$

Let the conjunction of these axioms over all transformations $T \xrightarrow{M} T^*$ in the M-calculus be

$$\ldots (2)_M$$

Finally, the axiom

$$\forall x\, \forall y\, \forall z (x \xrightarrow{M} y \,\&\, y \xrightarrow{M} z \Rightarrow x \xrightarrow{M} z) \quad . \; . \; . \quad (3)$$

allows deduction of relations which result from a sequence of substitutions.

It is clear that $(1)\,\&\,(2)_M\,\&\,(3) \Rightarrow W_1 \xrightarrow{M} W_2$ is a valid formula if and only if $W_1 \xrightarrow{M} W_2$ is true, so in particular the formula φ_M:

$$(1)\,\&\,(2)_M\,\&\,(3) \Rightarrow *\ulcorner M \urcorner * \xrightarrow{M} \diamond$$

is valid if and only if M on $\ulcorner M \urcorner$ eventually halts on $\square$.

Now given a machine M we can construct the formula φ_M, and assuming an algorithm for logical validity, decide whether φ_M is valid. This gives us an algorithm for the M on $\ulcorner M \urcorner$ halting problem, which is impossible, so in fact an algorithm for logical validity does not exist. In other words, predicate logic is undecidable (Turing, 1936; Church, 1936).

EXERCISE (see Post, 1947). *Construct an M-semigroup by replacing each transformation $T \to T^*$ by an equation $T = T^*$. Show that*

$$*\ulcorner M \urcorner * = \diamond \Leftrightarrow *\ulcorner M \urcorner * \xrightarrow{M} \diamond.$$

UNDECIDABILITY OF NUMBER THEORY

Again using the M-calculus, we shall show that for each Turing machine M we can construct a number theory formula ψ_M which is true if and only if M on $\ulcorner M \urcorner$ eventually halts on $\square$. Since the words are over the 6-letter alphabet $\{\square, 1, ', q, *, \diamond\}$, it is natural to regard this alphabet as $\{1,2,3,4,5,6\}$, and let words be numerals in base 7. Word transformations are then operations on numbers, and the relation $W_1 \xrightarrow{M} W_2$ is a certain number theoretic relation.

The difficulty is that numerals (to any base) are not particularly easy to represent in the language of 0, $'$, $+$ and $\cdot$. A second problem is the representation of *sequences* of substitutions. We are not able to introduce the transformation relation implicitly, as in logic, so the expedient of axiom (3) above is not open to us.

The necessary relations for the discussion of numerals are constructed as follows (notice that we are lucky to have a *prime* base, 7, for the third definition). We use $1, 2, 3, \ldots$ to abbreviate the terms $0', 0'', 0''', \ldots$

$$x < y \Leftrightarrow (\exists z)(\neg z = 0 \,\&\, x + z = y).$$

$$x \text{ divides } y \Leftrightarrow (\exists z)(\neg z = 0 \,\&\, \neg z = 1 \,\&\, x . z = y).$$

$$y \text{ is a power of } 7 \Leftrightarrow \forall x\, (x \text{ divides } y \Rightarrow 7 \text{ divides } x).$$

These are used to define the *concatenation* relation, which we use for building words.

$$\begin{aligned} x^\frown y = z &\Leftrightarrow \text{numeral for } z = \text{numeral for } x \text{ followed by the numeral for } y \\ &\Leftrightarrow z = xp+y \text{ where } p \text{ is the least power of 7 which exceeds } y \\ &\Leftrightarrow \exists p\ (p \text{ is a power of } 7\, \&\, z = xp+y\, \&\, y<p \\ &\qquad\qquad \&\, \forall q\ (q<p\, \&\, q \text{ a power of } 7 \Rightarrow q<y)). \end{aligned}$$

By substituting for the relations previously defined we finally get concatenation $^\frown$ explicitly defined in terms of $'$, $+$, $\cdot$.

Before representing $\xrightarrow{M}$ we have a separate relation $\mathrm{Tr}_1^M(x,y)$ for one-step transformation (resulting from a substitution) in the M-calculus. It is simply the conjunction of the formulas

$$\begin{aligned} \exists p\ \exists q((x = p^\frown T\, \&\, y = p^\frown T^*) \vee (x = T^\frown q\, \&\, y = T^{*\frown} q) \\ \vee (x = p^\frown T^\frown q\, \&\, y = p^\frown T^{*\frown} q)) \end{aligned}$$

over all transformations $T \xrightarrow{M} T^*$ in the M-calculus. (The words T, T^* of course now have $^\frown$ signs between their letters.)

As we noted, there is a difficulty in the fact that the general transformation relation, $\mathrm{Tr}^M(x,y)$, involves finite sequences of substitutions. However, since words in the M-calculus have no zeros, we can encode a sequence of substitutions $x \to x_1 \to x_2 \to \ldots \to x_n \to y$ by a single number

$$z = x^\frown 0^\frown x_1{}^\frown 0^\frown x_2 \ldots {}^\frown x_n{}^\frown 0^\frown y,$$

which is characterized by the following properties:

(i) x is the segment before the first 0.

(ii) y is the segment after the last 0.

(iii) If u, v are segments between successive zeros, then $\mathrm{Tr}_1^M(u,v)$.

We can talk about segments between zeros using the following property, definable using $^\frown$:

$$\begin{aligned} \mathrm{noz}(x) &\Leftrightarrow x \text{ has no zeros in its base 7 numeral} \\ &\Leftrightarrow \forall y\, \forall z\ (\neg\, x = y^\frown 0\, \&\, \neg\, x = y^\frown 0^\frown z) \end{aligned}$$

and then $\mathrm{Tr}^M(x,y)$ can be defined by saying there is a z with properties (i), (ii) and (iii), namely

$$\begin{aligned} \mathrm{Tr}^M(x,y) \Leftrightarrow (\exists z)(\exists p(z = x^\frown 0^\frown p\, \&\, \mathrm{noz}(x))\, \&\, \exists q(z = q^\frown 0^\frown y\, \&\, \mathrm{noz}(y)) \\ \&\, \forall s \forall t\, \forall u\, \forall v(z = s^\frown 0^\frown u^\frown 0^\frown v^\frown 0^\frown t \\ \&\, \mathrm{noz}(u) \&\, \mathrm{noz}(v) \Rightarrow \mathrm{Tr}_1^M(u,v))). \end{aligned}$$

This finally gives a formula ψ_M in the language of $0, ', +, \cdot$ which expresses $*\ulcorner M \urcorner * \xrightarrow{M} \Diamond$, namely $\mathrm{Tr}^M(5^\frown \ulcorner M \urcorner^\frown 5, 6)$.

As we argued in the case of predicate logic, an algorithm for deciding truth of formulas in number theory would yield one for the M on $\ulcorner M \urcorner$ halting problem, hence no such algorithm exists—number theory is undecidable (essentially due to Gödel, 1931).

Incompleteness

We define a *formal system* for some set Σ of sentences (given in some finite alphabet) to be a Turing machine which, when given some $\sigma \in \Sigma$, eventually halts on $\square$ only if σ is true. The formal system is *complete* when the "only if" is an "if and only if".

As we mentioned at the beginning of this section, conventional axiom systems can be shown to fit this definition, and indeed the system for predicate logic appears naturally in this form. The definition also has the technical advantage that to prove incompleteness we only need consider how a machine behaves on sentences of a particular form.

Now that we have found sentences ψ_M equivalent to "M on $\ulcorner M \urcorner$ eventually halts on $\square$", this leaves us with practically nothing to do but repeat the argument by which we established the unsolvability of the halting problem.

Theorem. *There is no complete formal system for number theory* (Gödel, 1931).

Proof. Let S be a formal system for number theory. Thus if S is given $\neg\psi_M$, S eventually halts on $\square$ only if $\neg\psi_M$ is true, that is, if M on $\ulcorner M \urcorner$ never halts on $\square$.

Now construct the machine $Z = \boxed{R} \rightarrow \boxed{S}$ where R is a machine which converts $\ulcorner M \urcorner$ to $\neg\psi_M$. We see that Z on $\ulcorner M \urcorner$ eventually halts on $\square$ only when M on $\ulcorner M \urcorner$ never halts on $\square$.

Putting $M = Z$, we see that Z on $\ulcorner Z \urcorner$ never halts on $\square$, otherwise there is a contradiction. So the sentence $\neg\psi_Z$ is *true*, but S does not "prove" it (since this means halting on $\square$).

Thus S is incomplete, and $\neg\psi_Z$ is a specific true sentence which S fails to prove.

Remarks. In Gödel's terminology, $\neg\psi_Z$ is a "sentence which asserts its own unprovability" since

$$\begin{aligned}\neg\psi_Z &\Leftrightarrow Z \text{ on } \ulcorner Z \urcorner \text{ never halts on } \square\\ &\Leftrightarrow S \text{ on } \neg\psi_Z \text{ never halts on } \square\\ &\Leftrightarrow S \text{ does not prove } \neg\psi_Z.\end{aligned}$$

One can see that a sentence which says "I am not provable" cannot indeed be provable without leading to contradiction. (Though being unprovable, it is true.) The bulk of the celebrated paper by Gödel (1931) is devoted to the construction of such a sentence within formal number theory.

References

E. Borel (1952), *Les Nombres Inaccessibles* (Gauthier-Villars).
A. Church (1936), "A note on the Entscheidungsproblem", *J. Symbolic Logic* **1**, 40–41.
J. Crossley and others (1952), *What is Mathematical Logic?* (Oxford University Press).
M. Davis (1965), *The Undecidable* (Raven Press).
K. Gödel (1930), "Die Vollständigkeit der Axiom des logischen Funktionenkalküls", *Monatsch. Math. Phys.* **37**, 349–360.
K. Gödel (1931), "Über formal unentscheidbare Sätze der Principia Mathematica und verwandte Systeme I", *Monatsh. Math. Phys.* **38**, 173–198.
J. Herbrand (1930), "Recherches sur la théorie de la demonstration", *Travaux de la Société des Lettres de Varsovie, Classe III, sciences math. et phys.*, **33**.
S. C. Kleene (1958), "Mathematical logic", *Proc. Int. Cong. Math.* 1958, 137–153.
L. Löwenheim (1915), "Über Möglichkeiten im Relativkalkül", *Math. Ann.* **76**, 447–470.
E. Post (1921), "Introduction to a general theory of elementary propositions", *Amer. J. Math.* **43**, 163–185.
E. Post (1944), "Recursively enumerable sets of positive integers and their decision problems", *Bull. Amer. Math. Soc.* **50**, 284–316.
E. Post (1947), "Recursive unsolvability of a problem of Thue", *J. Symb. Logic* **12**, 1–11.
T. Skolem (1955), *Mathematical Interpretation of Formal Systems* (North Holland).
R. Smullyan (1961), *Theory of Formal Systems* (Princeton University Press).
A. Turing (1936), "On computable numbers with an application to the Entscheidungsproblem", *Proc. Lond. Math. Soc.* (2), **42**, 230–265.

Monash University
Clayton 3168
Australia