

6. THE UNDECIDABLE

NOTE: This chapter has yet to be written in its full version. What follows is just an outline.

A statement, p , is *decidable* if either it, or its negation "not p " can be proved. Statements can be classified into Decidable and Undecidable as well as into True and False.

	TRUE	FALSE
DECIDABLE	PROVABLE	INVALID
UNDECIDABLE		

But do there exist undecidable statements?

Theorem: THIS STATEMENT CANNOT BE PROVED

Proof: If it is false it can be proved and so is true, a contradiction. Hence it must be true.

NOTE: We proved it true so it is provable after all!

This paradox is a version of the "this sentence is false" variety. It arises by blurring the distinction between statements within a system and meta-statements about the system.

At the beginning of the 20th century Hilbert proposed a number of major problems to be tackled during the century. One of them was:

"Design an algorithm that, given a formal axiomatic mathematical system, can determine whether a given statement is true or false."

FORMAL SYSTEM: Consists of an alphabet of symbols plus a number of rules which determine which strings are acceptable. ("Acceptable" may mean "true" or may simply mean "meaningful", that is "has correct syntax".)

For example in ordinary algebra the string " $(x+y)(x-y)$ " is meaningful but " $)+(xy+$ " is not.

EXAMPLE 1:

ALPHABET: variables x, y, \dots

other symbols $+, -, (,)$

RULES:(1) Each variable is an OK formula;

(2) If E, F are OK formulae so are $E + F, E - F$ and $(E)(F)$.

EXAMPLE 2:

ALPHABET: $-, T$

RULES:(1) "T" is an OK string;

(2) "TTT-" is an OK string;

(3) The string "TTTT" can be inserted or deleted anywhere;

(4) The string "--" can be inserted or deleted anywhere;

(5) The string "-TTT" can be replaced by "T-" anywhere.

We can consider this as an axiomatic system by considering acceptable strings as being true, "-" as denoting negation ("not"). Rules (1), (2) would be the "axioms" and rules (3) - (5) would be considered as the "inference rules".

Theorem: -TTT-

Proof: T by axiom 1;

T— by rule 4;

T-TTTT— by rule 3;

TT-T— by rule 5;

TT-TTTTT— by rule 3;

TTT-TT— by rule 5;

TTT-TTTTTT— by rule 3;

TTTT-TTT— by rule 5;

-TTT— by rule 3.

Note that both "TTT—" (axiom 2) and its negation "-TTT—" are theorems. This means that this is an inconsistent system.

EXAMPLE 3:

As example 3 but with axiom 2 deleted. The system can now be shown to be consistent. No contradiction can ever be deduced. That is, we can never prove both a theorem and its negation.

Example 3 is "consistent" but is not "complete". That is, there are statements which can not be proved, but nor can their negation. They are "undecidable" statements.

"T-T" is undecidable. This is because the single axiom has an odd number of T's and each of the inference rules add or subtract an even number to the number of T's. Hence all theorems will have an odd number of T's and so neither "T-T" nor its negation "-T-T" are theorems.

EXAMPLE 4:

In the following system we have a particular interpretation in mind but we must refrain from attaching any meaning to the words "father", "mother" etc and to use any knowledge other than that which is explicitly stated in the axioms.

Imagine communicating with angels who have no notion of parents. We can't take anything for granted. Every fact which seems "self-evident" to us, such as the impossibility of being your own father, must be either explicitly stated as an axiom or be deducible from the axioms.

This system could be expressed in symbols, and indeed we would do this if we were setting it up on a computer, but we shall be more informal.

Let us define a *family* to be a set T with two undefined relations *father of* and *mother of*. We call the elements of T *persons* and define the relation *parent of* to mean "father or mother of". Define *ancestor* by:

- (1) Every parent of x is an ancestor of x;
- (2) Every parent of an ancestor of x is an ancestor of x.

AXIOMS:

Axiom 1: Each person has a unique mother and a unique father.

Axiom 2: No father can be a mother.

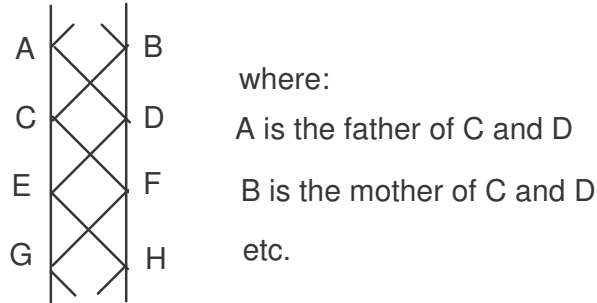
Axiom 3: No person can be his/her own ancestor.

Our intended interpretation is not the only model which fit these axioms.

MODEL 1: F is a set of people and *father, mother* etc have their usual interpretations.

MODEL 2: $F = \{1,2,3,\dots\}$; *father of x* means $2x+1$; *mother of x* means $2x$.

MODEL 3:



THE SYSTEM OF ARITHMETIC

Gödel No.	Symbol	Meaning	Examples
0	0	0	zero
1	\	successor	$2 = \backslash 0$
2	x	variable	$x, \backslash x, \backslash \backslash x, \dots$
3	S	for some	"m is even" is $S \backslash x E x T \backslash \backslash 0 \backslash x$
4	A	for all	
5	I	implies	" $(x + 1 = 2) \Rightarrow (x = 1)$ " is $A x I E P x \backslash 0 \backslash \backslash 0 E x \backslash 0$
6	P	plus	
7	T	times	" $2x + 1$ " is $P T \backslash \backslash 0 x \backslash 0$
8	E	equals	
9	N	not	" $x \neq 0$ " is $N E x 0$

This version of arithmetic avoids the need for brackets by using what is known as the "reverse Polish convention". We read from left to right until we reach a point where a segment is completed.

The standard logical constructions can be expressed in this language. (In the following the symbols "p" and "q" are not variables in the language. Rather, they stand for expressions in the language that represent statements.)

"p or q" is "(not p) implies q" i.e. $I N p q$

"p and q" is "not(p implies not q)" i.e. $N I p N q$

For example " $x = 1$ " is $E x \backslash 0$ and " $x = 2$ " is $E x \backslash \backslash 0$ so

" $x = 1$ or $x = 2$ " is $I N E x \backslash 0 E x \backslash \backslash 0$ and

" $x = 1$ and $x = 2$ " is $N I E x \backslash \backslash 0 N E x \backslash 0$.

There are other constructions in arithmetic than just addition and multiplication. But these can all be expressed in more primitive ways within the language.

For example " $x \leq y$ " means "there is some z with $x + z = y$ " and so can be expressed as $S \backslash \backslash x E P x \backslash \backslash x$.

The statement "x divides y", or in other words "y is a multiple of x", means there is some quotient q such that $y = xq$. It can therefore be expressed as $\exists x \exists y \exists q (y = xq)$.

The expressions representing even fairly simple statements can end up being very complex. The statement "p is prime" becomes $\exists x \exists y \exists z (x = yz \wedge \neg \exists d (d \mid x \wedge d \neq 1 \wedge d \neq x))$. Nevertheless there are only two reasons for doing this sort of thing and in neither of them is complexity particularly significant.

(i) If we wish to program a computer to prove theorems, an incomprehensible string such as the one above can be readily processed.

More importantly for us here, the question is not so much *how* we can represent various arithmetic statements but *whether* we can. And all of arithmetic can be expressed using only these 10 symbols.

AXIOMS: Various sets of axioms have been proposed. Gödel showed that none of them can possibly lead to a complete axiomatic system. We will always have undecidable statements no matter how many extra axioms we add.

RULES OF INFERENCE: Usually only the following two rules of inference are used. But Gödel showed that even if we added more rules it would not change the incompleteness of arithmetic.

Rule of Substitution: Any expression representing a number can be substituted for any variable in a theorem.

Modus Ponens: If statements of the form $\mathbf{I}pq$ and \mathbf{p} have been proved, the statement \mathbf{q} is also a theorem.

GÖDEL NUMBERS

Example:

If $\alpha = \exists x \exists y \exists z (x = yz)$ (there is no predecessor to 0) is coded as

$\Gamma(\alpha) = 932862100$, called the Gödel number of the statement.

If Π is the list of statements L_1, L_2, \dots, L_k (e.g. a proof), its Gödel number is:

$$2^{\Gamma(L_1)} \cdot 3^{\Gamma(L_2)} \cdot \dots \cdot p_k^{\Gamma(L_k)}$$

where p_k is the k'th prime.

Since every number can be factorised uniquely into primes we can always recover the sequence from its Gödel Number.

The brilliant discovery of Gödel was the fact that it is possible to express meta-mathematical statements about this system as arithmetic statements *within* the system.

For example we can express the statement that "the last symbol in the expression whose Gödel number is N, is a variable" by "10 divides $N-2$ ".

The meta-mathematical statement: "the statement with Gödel number N is the first line of some proof with Gödel number M" is coded as " $M = 2^N$ times some odd number".

The meta-mathematical statement $P(M,N)$ that "the sequence of statements with Gödel number M is a proof for the statement with Gödel number N" can be expressed arithmetically. It involves not only checking that the statement is the last line of the proof but also that the proof is correct in the sense that each line can be justified by the axioms and rules of inference. But since this check could be performed by a computer program, and our language includes all the necessary logic to mimic any such program, such a program could be translated as an arithmetic statement.

Then there is the question of consistency. Nobody has been able to construct a set of axioms for arithmetic, rich enough to do all the arithmetic we are familiar with, which has been proved to be consistent! Quite possibly no such set of axioms exists!

There appears to be a fundamental dilemma that we must face. We want to put our mathematics on a firm foundation and to be able to guarantee that everything we prove is true and everything that is true can be proved. But instead we are doomed to do our arithmetic by faith, *believing* that no unresolvable contradiction will ever arise but without the assurance that this is indeed so. The only alternatives are safe but totally useless arithmetics. It seems we must live dangerously.

Computer scientists face a similar dilemma. Either use a safe, but totally useless, programming language or a language powerful enough to be both useful and dangerous.