Algorithms, Equations, and Logic.

Martin Davis

Professor Emeritus
Courant Institute, NYU

Visiting Scholar
UC Berkeley

The Plot

Our discussion will be framed in terms of the natural numbers 0,1,2,3,…

We'll look at three different ways to specify a set of natural numbers:

1. By an algorithm to *decide* membership in the set.
2. By an algorithm that *lists* all the members of the set.
3. As the parameter values for which an *equation* has solutions.

This will yield remarkable results.

Example: The set of even numbers $\{0, 2, 4, \ldots\}$

Algorithm to *decide* membership in this set:

```
Input n
Divide n by 2
Let q be the quotient and r the remainder
If r = 0 then Output ``Yes''
         else Output ``No''
```

Example: The set of even numbers $\{0, 2, 4, \ldots\}$

Algorithm to *decide* membership in this set:

```
Input n
Divide n by 2
Let q be the quotient and r the remainder
If r = 0 then Output ``Yes''
         else Output ``No''
```

Algorithm to *list* the members of this set

$$k \leftarrow 0$$

```
Repeat forever {Output k
```

$$k \leftarrow k + 2\}$$

Example: The set of even numbers $\{0, 2, 4, \ldots\}$

Algorithm to *decide* membership in this set:

```
Input n
Divide n by 2
Let q be the quotient and r the remainder
If r = 0 then Output ``Yes''
         else Output ``No''
```

Algorithm to *list* the members of this set

$$k \leftarrow 0$$

```
Repeat forever {Output k
```

$$k \leftarrow k + 2\}$$

Equation to specify the set as parametric values

$$a - 2x = 0$$

Example: The set of even numbers $\{0, 2, 4, \ldots\}$

Algorithm to *decide* membership in this set:

```
Input n
Divide n by 2
Let q be the quotient and r the remainder
If r = 0 then Output ``Yes''
         else Output ``No''
```

Algorithm to *list* the members of this set

$$k \leftarrow 0$$

```
Repeat forever {Output k
```

$$k \leftarrow k + 2\}$$

Equation to specify the set as parametric values

$$a - 2x = 0$$

- A set of natural numbers is **decidable** if there is an algorithm that decides membership in it.
- A set of natural numbers is **listable** if there is an algorithm that lists its members (in any order with repetitions permitted).
- A set of natural numbers is **Diophantine** if there is a polynomial equation $p(a, x_1, \ldots, x_\ell) = 0$ with integer coefficients which has natural number solutions for exactly those values of a that are members of the set.

Every decidable set is listable

Let $\mathcal{A}$ be an algorithm that *decides* membership in a set S. The following algorithm lists the members of S:

```
n ← 0
repeat forever
{Input n to 𝒜
if 𝒜 Outputs ``Yes'' then OUTPUT n
n ← n + 1}
```

The *complement* of a set S, written $\bar{S}$, is the set of all natural numbers that don't belong to S.

If $S, \bar{S}$ are both listable, then S is decidable

Let $\mathcal{A}, \mathcal{B}$ be algorithms that list S and $\bar{S}$, respectively. The following algorithm decides membership in S:

```
Input n
N ← 100
{Run 𝒜 and ℬ for N steps
If 𝒜 outputs n, then {OUTPUT ``Yes''
                      STOP}
If ℬ outputs n, then {OUTPUT ``No''
                      STOP}}
N ← N + 100
```

Examples of Diophantine Sets

- $a - (x+2)(y+2) = 0$ specifies the set of *composite numbers*.
- $a - (2x+3)(y+1) = 0$ specifies the set of numbers not powers of 2.
- The "Pell" equation $x^2 - a(y+1)^2 - 1 = 0$ *specifies* the set consisting of 0 and the numbers not perfect squares.

Examples of Diophantine Sets

- $a-(x+2)(y+2)=0$ specifies the set of *composite numbers*.
- $a-(2x+3)(y+1)=0$ specifies the set of numbers not powers of 2.
- The "Pell" equation $x^2-a(y+1)^2-1=0$ *specifies* the set consisting of 0 and the numbers not perfect squares.

Every Diophantine set is listable

The following algorithm lists the set specified by $p(a,x_1,\ldots,x_\ell)=0$:

$N \leftarrow 100$
`repeat forever`
$\{$`for` $a,x_1,\ldots,x_\ell \leq N$ `do`
$\{\ r \leftarrow p(a,x_1,\ldots,x_\ell)$
`if` $r=0$ `then Output` $a\}$
$N \leftarrow N+100\}$

Examples of Diophantine Sets

- $a-(x+2)(y+2)=0$ specifies the set of *composite numbers*.
- $a-(2x+3)(y+1)=0$ specifies the set of numbers not powers of 2.
- The "Pell" equation $x^2-a(y+1)^2-1=0$ *specifies* the set consisting of 0 and the numbers not perfect squares.

Every Diophantine set is listable

The following algorithm lists the set specified by $p(a,x_1,\ldots,x_\ell)=0$:

$N \leftarrow 100$
`repeat forever`
$\{$`for` $a,x_1,\ldots,x_\ell \leq N$ `do`
$\{\ r \leftarrow p(a,x_1,\ldots,x_\ell)$
`if` $r=0$ `then Output` $a\}$
$N \leftarrow N+100\}$

Is every listable set Diophantine?

TWO THEOREMS

Unsolvability Theorem: ***There is a listable set K whose complement $\bar{K}$ is not listable. Therefore K is not decidable.***
Proof. Later if time permits

MRDP Theorem: ***If a set is listable, then it is also Diophantine.***

TWO THEOREMS

Unsolvability Theorem: ***There is a listable set K whose complement $\bar{K}$ is not listable. Therefore K is not decidable.***
Proof. Later if time permits

MRDP Theorem: ***If a set is listable, then it is also Diophantine.***

Corollary: ***There is a polynomial*** $p_o(a, x_1, \ldots, x_\ell)$ ***such that the equation***

$$p_o(a, x_1, \ldots, x_\ell) = 0$$

specifies the set K. Hence, no algorithm exists to determine of a given value of a whether or not there exist natural numbers $x_1, \ldots, x_\ell$ that satisfy this equation.

TWO THEOREMS

Unsolvability Theorem: ***There is a listable set K whose complement $\bar{K}$ is not listable. Therefore K is not decidable.***
Proof. Later if time permits

MRDP Theorem: ***If a set is listable, then it is also Diophantine.***

Corollary: ***There is a polynomial*** $p_o(a, x_1, \ldots, x_\ell)$ ***such that the equation***

$$p_o(a, x_1, \ldots, x_\ell) = 0$$

specifies the set K. Hence, no algorithm exists to determine of a given value of a whether or not there exist natural numbers $x_1, \ldots, x_\ell$ that satisfy this equation.

UNSOLVABILITY OF HILBERT'S 10TH PROBLEM
Corollary:***No algorithm exists to determine of a given polynomial equation with integer coefficients whether or not it has a solution in natural numbers.***

LOGIC

A *formal logical system* provides

- a special language in which propositions are represented by strings of symbols
- a list of initial strings or "axioms"
- rules of inference for obtaining new strings from given strings

The strings thus obtained are called the *theorems* of the system. From our point of view a formal logical system provides an algorithm that makes a list of its theorems.

***110·643.** $\vdash . 1 +_c 1 = 2$

Dem.

$$\vdash . *110{\cdot}632 . *101{\cdot}21{\cdot}28 . \supset$$
$$\vdash . 1 +_c 1 = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota`y \in 1\}$$
$$[*54{\cdot}3] \quad = 2 . \supset \vdash . \text{Prop}$$

The above proposition is occasionally useful. It is used at least three times, in *113·66 and *120·123·472.

*110·7·71 are required for proving *110·72, and *110·72 is used in *117·3, which is a fundamental proposition in the theory of greater and less.

***110·7.** $\vdash : \beta \subset \alpha . \supset . (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu$

Dem.

$$\vdash . *24{\cdot}411{\cdot}21 . \supset \vdash : \text{Hp} . \supset . \alpha = \beta \cup (\alpha - \beta) . \beta \cap (\alpha - \beta) = \Lambda .$$
$$[*110{\cdot}32] \qquad \supset . \text{Nc}`\alpha = \text{Nc}`\beta +_c \text{Nc}`(\alpha - \beta) : \supset \vdash . \text{Prop}$$

***110·71.** $\vdash : (\exists \mu) . \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu . \supset . (\exists \delta) . \delta \text{ sm } \beta . \delta \subset \alpha$

Dem.

$\vdash . *100{\cdot}3 . *110{\cdot}4 . \supset$

$\vdash : \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu . \supset . \mu \in \text{NC} - \iota`\Lambda$ (1)

$\vdash . *110{\cdot}3 . \supset \vdash : \text{Nc}`\alpha = \text{Nc}`\beta +_c \text{Nc}`\gamma . \equiv . \text{Nc}`\alpha = \text{Nc}`(\beta + \gamma) .$

$[*100{\cdot}3{\cdot}31] \qquad \supset . \alpha \text{ sm } (\beta + \gamma) .$

$[*73{\cdot}1] \qquad \supset . (\exists R) . R \in 1 \rightarrow 1 . \text{D}`R = \alpha . \text{Cl}`R = \downarrow \Lambda_\gamma``\iota``\beta \cup \Lambda_\beta \downarrow``\iota``\gamma .$

$[*37{\cdot}15] \qquad \supset . (\exists R) . R \in 1 \rightarrow 1 . \downarrow \Lambda_\gamma``\iota``\beta \subset \text{Cl}`R . R`` \downarrow \Lambda_\gamma``\iota``\beta \subset \alpha .$

$[*110{\cdot}12 . *73{\cdot}22] \supset . (\exists \delta) . \delta \subset \alpha . \delta \text{ sm } \beta$ (2)

$\vdash . (1) . (2) . \supset \vdash . \text{Prop}$

The above proof depends upon the fact that "$\text{Nc}`\alpha$" and "$\text{Nc}`\beta +_c \mu$" are typically ambiguous, and therefore, when they are asserted to be equal, this must hold in *any* type, and therefore, in particular, in that type for which we have $\alpha \in \text{Nc}`\alpha$, *i.e.* for $\text{N}_0\text{c}`\alpha$. This is why the use of *100·3 is legitimate.

***110·72.** $\vdash : (\exists \delta) . \delta \text{ sm } \beta . \delta \subset \alpha . \equiv . (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu$

Dem.

$\vdash . *100{\cdot}321 . *110{\cdot}7 . \supset$

$\vdash :. \delta \text{ sm } \beta . \delta \subset \alpha . \supset : \text{Nc}`\delta = \text{Nc}`\beta : (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\delta +_c \mu :$

$[*13{\cdot}12] \qquad \supset : (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu$ (1)

$\vdash . (1) . *110{\cdot}71 . \supset \vdash . \text{Prop}$

LOGIC (continued)

Let $\mathcal{L}$ be a particular formal system that, for each $a = 0, 1, 2, \ldots$ uses a string we'll call Π_a to represent the proposition: **The equation**

$$p_o(a, x_1, \ldots, x_\ell) = 0 \tag{1}$$

has no solutions in natural numbers $x_1, \ldots, x_\ell$**.**

[This proposition is equivalent to saying that $a \in \bar{K}$.]

We say that $\mathcal{L}$ is *sound* if whenever a string Π_a is a theorem of $\mathcal{L}$, the proposition that it represents is true.

LOGIC (continued)

Let $\mathcal{L}$ be a particular formal system that, for each $a = 0, 1, 2, \ldots$ uses a string we'll call Π_a to represent the proposition: **The equation**

$$p_o(a, x_1, \ldots, x_\ell) = 0 \tag{1}$$

has no solutions in natural numbers $x_1, \ldots, x_\ell$**.**

[This proposition is equivalent to saying that $a \in \bar{K}$.]

We say that $\mathcal{L}$ is *sound* if whenever a string Π_a is a theorem of $\mathcal{L}$, the proposition that it represents is true.

Gödel Incompleteness Theorem. ***Let $\mathcal{L}$ be sound. Then there is a number a for which equation*** (1) ***has no solutions in natural numbers although Π_a is not a theorem of $\mathcal{L}$.***

Proof: Otherwise, it would be the case that $a \in \bar{K}$ if and only if Π_a is a theorem of $\mathcal{L}$. So by listing the theorems of $\mathcal{L}$ and selecting the Π_a as they appear, it would be possible to list the members of $\bar{K}$. But this is impossible because $\bar{K}$ is not listable.

Thus for every *sound* logic, there is a true proposition not provable in that logic!

LOGIC and ROGER PENROSE

$$p_o(a, x_1, \ldots, x_\ell) = 0 \tag{1}$$

Gödel Incompleteness Theorem. ***Let $\mathcal{L}$ be sound. Then there is a number a for which equation (1) has no solutions in natural numbers although Π_a is not a theorem of $\mathcal{L}$.***

Thus for every *sound* logic, there is a true proposition not provable in that logic!

Roger Penrose claims that **we can see** that this proposition is true whereas a programmed computer, no matter how powerful won't have this ability. Therefore, he argues:

Our minds surpass any mere mechanism.

BUT: what we can truly **see** is that **IF** the logic is **sound**, then the proposition is true, and a suitably programmed computer can **see** exactly the same thing.

For logics satisfying some additional conditions, the hypothesis of soundness can be replaced by the weaker condition of **consistency**, meaning that no pair of propositions that contradict one another should both be provable. But to **see** that even this weaker condition holds can be very difficult to verify.

A PROOF

Unsolvability Theorem: ***There is a listable set K whose complement $\bar{K}$ is not listable. Therefore K is not decidable.***

Proof: Let us fix a particular programming language in which to write algorithms for listing the members of a set. Then all possible programs in that language can be written in a sequence:

$$\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$$

For $i = 0, 1, 2, \ldots$, let S_i be the set listed by $\mathcal{P}_i$.

Now, let K consist of those numbers i such that $i \in S_i$. Then we claim:

- K **is listable.** For each $n = 1, 2, 3, \ldots$ run each of the programs $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_n$ for n steps. Make a list as follows: Whenever a particular program $\mathcal{P}_i$ outputs the very number i, put i on the list.

- $\bar{K}$ **is not listable.** Suppose that $\bar{K}$ is listed by program $\mathcal{P}_{i_0}$. We ask: is $i_0 \in \bar{K}$? If so, it would be listed by $\mathcal{P}_{i_0}$, and hence would be in K. Contradiction. So i_0 must be in K. By definition, $i_0 \in S_{i_0}$, i.e., i_0 is listed by $\mathcal{P}_{i_0}$. But then $i_0 \in \bar{K}$. Again, a contradiction.