Arithmetisation of computation

Christopher J. Taylor

Recap

Last time Tomasz spoke of computable functions and computable sets.

- A set is computably enumerable if some algorithmic procedure can list its elements one-by-one.
- A set is computable if both it and its complement are computably enumerable.

To make this precise, a model of computation must be chosen.

Overview

Today's talk (5.1–5.7):

- The arithmetical hierarchy
- Smullyan's elementary formal systems (EFS)
- Turing machines and universal machines
- Implementing Peano arithmetic in an EFS
- Building up to Σ^0_1 formulas

Overview

Today's talk (5.1–5.7):

- The arithmetical hierarchy
- Smullyan's elementary formal systems (EFS)
- Turing machines and universal machines
- Implementing Peano arithmetic in an EFS
- Building up to Σ_1^0 formulas

Main result

Every Σ_1^0 formula can be realised by an EFS.

Overview

Today's talk (5.1–5.7):

- The arithmetical hierarchy
- Smullyan's elementary formal systems (EFS)
- Turing machines and universal machines
- Implementing Peano arithmetic in an EFS
- Building up to Σ^0_1 formulas

Main result

Every Σ^0_1 formula can be realised by an EFS.

- Bonus content: Hilbert's 10th problem

Overview

Main result

Every Σ_1^0 formula can be realised by an EFS.

Not today's talk (5.8–5.10):

- Arithmetising elementary formal systems: given an EFS, there is a Σ_1 formula realised by it.

Overview

Main result

Every Σ_1^0 formula can be realised by an EFS.

Not today's talk (5.8–5.10):

- Arithmetising elementary formal systems: given an EFS, there is a Σ_1 formula realised by it.
- Arithmetising computable enumeration: given a computably enumerable set S, there is a computable function f whose range is S
  - Then $f(0), f(1), f(2), \ldots$ is a proper list of S that can be computed.

Overview

Main result

Every Σ^0_1 formula can be realised by an EFS.

Not today's talk (5.8–5.10):

- Arithmetising elementary formal systems: given an EFS, there is a Σ_1 formula realised by it.
- Arithmetising computable enumeration: given a computably enumerable set S, there is a computable function f whose range is S
  - Then $f(0), f(1), f(2), \ldots$ is a proper list of S that can be computed.
- Arithmetising computable analysis: the definition of RCA_0.

The Arithmetical Heirarchy

Formulas can be classified depending on the level of alternation in their quantifiers:

$$(\forall x_1)(\exists x_2)(\forall x_3)(\forall x_4)(\exists x_5)(\forall x_6)\ \varphi(x_1, \ldots, x_6)$$

The formula above is Π_5 – the quantification is "5-fold" and it begins with $\forall$.

The Arithmetical Heirarchy

Formulas can be classified depending on the level of alternation in their quantifiers:

$$(\forall x_1)(\exists x_2)(\forall x_3)(\forall x_4)(\exists x_5)(\forall x_6)\ \varphi(x_1, \ldots, x_6)$$

The formula above is Π_5 – the quantification is "5-fold" and it begins with $\forall$.

Given a base case $\Sigma_0 = \Pi_0$, the arithmetical hierarchy is defined inductively:

- if φ is Π_n, then $(\exists x_1)(\exists x_2)\ldots(\exists x_k)\ \varphi$ is Σ_{n+1}

The Arithmetical Heirarchy

Formulas can be classified depending on the level of alternation in their quantifiers:

$$(\forall x_1)(\exists x_2)(\forall x_3)(\forall x_4)(\exists x_5)(\forall x_6)\ \varphi(x_1, \ldots, x_6)$$

The formula above is Π_5 – the quantification is "5-fold" and it begins with $\forall$.

Given a base case $\Sigma_0 = \Pi_0$, the arithmetical hierarchy is defined inductively:

- if φ is Π_n, then $(\exists x_1)(\exists x_2)\ldots(\exists x_k)\ \varphi$ is Σ_{n+1}
- if φ is Σ_n, then $(\forall x_1)(\forall x_2)\ldots(\forall x_k)\ \varphi$ is Π_{n+1}

The Arithmetical Heirarchy

Formulas can be classified depending on the level of alternation in their quantifiers:

$$(\forall x_1)(\exists x_2)(\forall x_3)(\forall x_4)(\exists x_5)(\forall x_6)\ \varphi(x_1, \ldots, x_6)$$

The formula above is Π_5 – the quantification is "5-fold" and it begins with $\forall$.

Given a base case $\Sigma_0 = \Pi_0$, the arithmetical hierarchy is defined inductively:

- if φ is Π_n, then $(\exists x_1)(\exists x_2)\ldots(\exists x_k)\ \varphi$ is Σ_{n+1}
- if φ is Σ_n, then $(\forall x_1)(\forall x_2)\ldots(\forall x_k)\ \varphi$ is Π_{n+1}

Then close the classes under logical equivalence.

The Arithmetical Heirarchy

Formulas can be classified depending on the level of alternation in their quantifiers:

$$(\forall x_1)(\exists x_2)(\forall x_3)(\forall x_4)(\exists x_5)(\forall x_6)\ \varphi(x_1, \ldots, x_6)$$

The formula above is Π_5 – the quantification is "5-fold" and it begins with $\forall$.

Given a base case $\Sigma_0 = \Pi_0$, the arithmetical hierarchy is defined inductively:

- if φ is Π_n, then $(\exists x_1)(\exists x_2)\ldots(\exists x_k)\ \varphi$ is Σ_{n+1}
- if φ is Σ_n, then $(\forall x_1)(\forall x_2)\ldots(\forall x_k)\ \varphi$ is Π_{n+1}

Then close the classes under logical equivalence.

Intuition: Σ_1 formulas can be *verified* by brute force. Π_1 formulas can be *falsified* by brute force.

The Arithmetical Heirarchy

Consider the following formulas (in the language of PA):

- $1 + 3 = 3 + 1$,
- $x(y + z) = xy + z$
- $x^2 + 1 = 0$
- $x \neq 1$ and $xy = z$

Note that these items are not quantified. Given *specific* values of x, y, z, both LHS and RHS can be evaluated and truthhood can be determined for those specific values.

The Arithmetical Heirarchy

Consider the following formulas (in the language of PA):

- $1 + 3 = 3 + 1$,
- $x(y + z) = xy + z$
- $x^2 + 1 = 0$
- $x \neq 1$ and $xy = z$

Note that these items are not quantified. Given *specific* values of x, y, z, both LHS and RHS can be evaluated and truthhood can be determined for those specific values.

These are examples of $\Sigma_0 = \Pi_0$ formulas. The idea is that they are at least on a surface level decidable.

The Arithmetical Heirarchy

Consider now the statement "*x* is prime", or

- $x \neq 0$ and $x \neq 1$ and $(\forall y)(\forall z)\ yz = x \implies (y = x \text{ or } z = x)$

The Arithmetical Heirarchy

Consider now the statement "x is prime", or

- $x \neq 0$ and $x \neq 1$ and $(\forall y)(\forall z)\ yz = x \implies (y = x$ or $z = x)$

Despite the quantifiers, primality of x *is* decidable[1] — the factors are bounded above by x, so there is a finite search space.

[1] polynomial, in fact, by the AKS primality test (2002)

The Arithmetical Heirarchy

Consider now the statement "x is prime", or

- $x \neq 0$ and $x \neq 1$ and $(\forall y)(\forall z)\ yz = x \implies (y = x \text{ or } z = x)$

Despite the quantifiers, primality of x *is* decidable[1] — the factors are bounded above by x, so there is a finite search space.

Both $(\forall x < y)$ and $(\exists x < y)$ are called *bounded quantifiers*, and a formula whose only quantification is bounded is classified as both Π_0 and Σ_0.

[1] polynomial, in fact, by the AKS primality test (2002)

The Arithmetical Heirarchy

Consider now the statement “x is prime”, or

- $x \neq 0$ and $x \neq 1$ and $(\forall y)(\forall z)\ yz = x \implies (y = x$ or $z = x)$

Despite the quantifiers, primality of x *is* decidable[1] — the factors are bounded above by x, so there is a finite search space.

Both $(\forall x < y)$ and $(\exists x < y)$ are called *bounded quantifiers*, and a formula whose only quantification is bounded is classified as both Π_0 and Σ_0.

Earlier in the book Stillwell defines Σ_0 and Π_0 as the set of *quantifier-free* formulas and then later redefines them in terms of bounded quantifiers.

[1] polynomial, in fact, by the AKS primality test (2002)

The Arithmetical Heirarchy

Consider now the statement "x is prime", or

- $x \neq 0$ and $x \neq 1$ and $(\forall y)(\forall z)\ yz = x \implies (y = x \text{ or } z = x)$

Despite the quantifiers, primality of x *is* decidable[1] — the factors are bounded above by x, so there is a finite search space.

Both $(\forall x < y)$ and $(\exists x < y)$ are called *bounded quantifiers*, and a formula whose only quantification is bounded is classified as both Π_0 and Σ_0.

Earlier in the book Stillwell defines Σ_0 and Π_0 as the set of *quantifier-free* formulas and then later redefines them in terms of bounded quantifiers.

Whichever definition is chosen, Σ_i and Π_i are unchanged for $i > 0$.

[1] polynomial, in fact, by the AKS primality test (2002)

Smullyan's elementary formal systems

An elementary formal system is a collection of the following items:

1. a finite alphabet $A = \{a, b, c, \dots\}$ not containing $\Rightarrow$,
2. a set of variables $V = \{x, y, z, \dots\}$ disjoint from A,
3. a set of set variables $S = \{P, Q, R, \dots\}$ disjoint from $A \cup V$,
4. axioms of the form Pw, with $P \in S$ and $w \in (A \cup V)^*$,
5. axioms of the form $P_1 x_1 \Rightarrow P_2 x_2 \Rightarrow \dots \Rightarrow P_n x_n$, where each P_i is a set variable and each x_i is a word in $(A \cup V)^*$.

Smullyan's elementary formal systems

An elementary formal system is a collection of the following items:

1. a finite alphabet $A = \{a, b, c, \dots\}$ not containing $\Rightarrow$,
2. a set of variables $V = \{x, y, z, \dots\}$ disjoint from A,
3. a set of set variables $S = \{P, Q, R, \dots\}$ disjoint from $A \cup V$,
4. axioms of the form Pw, with $P \in S$ and $w \in (A \cup V)^*$,
5. axioms of the form $P_1x_1 \Rightarrow P_2x_2 \Rightarrow \dots \Rightarrow P_nx_n$, where each P_i is a set variable and each x_i is a word in $(A \cup V)^*$.

Note:

- Everything is finite.
- Arbitrary words in A^* will be substituted for variables.
- $P_1x_1 \Rightarrow \dots \Rightarrow P_nx_2$ is interpreted as $(P_1x_1 \wedge \cdots \wedge P_{n-1}x_{n-1}) \Rightarrow P_n$.
- There are no brackets.

Smullyan's elementary formal systems

An EFS generating the set E of strings of the form $aa \ldots a$ of even positive length is

$$Eaa$$

$$Ex \Rightarrow Exaa$$

Smullyan's elementary formal systems

An EFS generating the set E of strings of the form $aa \dots a$ of even positive length is

$$Eaa$$
$$Ex \Rightarrow Exaa$$

For any given EFS, the rules of inference are as follows:

- For any axiom, substituting an arbitrary word for each variable in that axiom gives a theorem.
- If U and $U \Rightarrow V$ are theorems, and U is not itself of the form $X \Rightarrow Y$, then V is a theorem.

Smullyan's elementary formal systems

An EFS generating the set E of strings of the form $aa \ldots a$ of even positive length is

$$Eaa$$

$$Ex \Rightarrow Exaa$$

For any given EFS, the rules of inference are as follows:

- For any axiom, substituting an arbitrary word for each variable in that axiom gives a theorem.
- If U and $U \Rightarrow V$ are theorems, and U is not itself of the form $X \Rightarrow Y$, then V is a theorem.

To justify the use of elementary formal systems we will refer to an excerpt from the book.

Smullyan's elementary formal systems

An EFS generating the set P of non-empty palindromes on the alphabet $\{a, b\}$:

$$Pa$$
$$Paa$$
$$Pb$$
$$Pbb$$
$$Px \Rightarrow Paxa$$
$$Px \Rightarrow Pbxb$$

Smullyan's elementary formal systems

An EFS generating the set P of non-empty palindromes on the alphabet $\{a, b\}$:

$$Pa$$
$$Paa$$
$$Pb$$
$$Pbb$$
$$Px \Rightarrow Paxa$$
$$Px \Rightarrow Pbxb$$

For relations, we assume the comma symbol is not in the alphabet, add it, and then permit rules of the form $Px_1, \ldots, x_n \Rightarrow Qy_1, \ldots, y_n$.

Smullyan's elementary formal systems

An EFS generating the set P of non-empty palindromes on the alphabet $\{a, b\}$:

$$Pa$$
$$Paa$$
$$Pb$$
$$Pbb$$
$$Px \Rightarrow Paxa$$
$$Px \Rightarrow Pbxb$$

For relations, we assume the comma symbol is not in the alphabet, add it, and then permit rules of the form $Px_1, \ldots, x_n \Rightarrow Qy_1, \ldots, y_n$. Suppose we amended the above EFS to include

$$Px \Rightarrow Py \Rightarrow Sx, y.$$

Then Sx, y is a theorem if and only if (x, y) is an ordered pair of palindromes.

Implementing PA

Stillwell says of base 1 numerals, “These *base one* or *unary* numerals are simple and natural, but in some ways too simple to be convenient.”

Implementing PA

Stillwell says of base 1 numerals, “These *base one* or *unary* numerals are simple and natural, but in some ways too simple to be convenient.”

Instead, we utilise what Smullyan calls the *dyadic* system of numerals.

$$
\begin{aligned}
1 &= 1 \\
2 &= 2 \\
3 &= 11 \\
4 &= 12 \\
5 &= 21 \\
6 &= 22 \\
7 &= 111 \\
&\cdots
\end{aligned}
$$

Implementing PA

Stillwell says of base 1 numerals, “These *base one* or *unary* numerals are simple and natural, but in some ways too simple to be convenient.”

Instead, we utilise what Smullyan calls the *dyadic* system of numerals.

$$1 = 1$$
$$2 = 2$$
$$3 = 11$$
$$4 = 12$$
$$5 = 21$$
$$6 = 22$$
$$7 = 111$$
$$\dots$$

In general, a positive integer n is represented by a string $d_k \dots d_2 d_1$, where

$$n = d_k 2^{k-1} + \cdots + d_2 \cdot 2 + d_1 \cdot 1.$$

Implementing PA

The goal now is to build an EFS for each of the basic relations of PA,

1. $S(x) = y$,
2. $x + y = z$,
3. $x \cdot y = z$,
4. $x < y$,
5. $x \leqslant y$,
6. $x \neq y$

For an EFS to encode the relation $S(x) = y$, we mean that for some set variable P, the EFS proves Px, y if and only if $S(x) = y$.

Implementing PA

The goal now is to build an EFS for each of the basic relations of PA,

1. $S(x) = y$,
2. $x + y = z$,
3. $x \cdot y = z$,
4. $x < y$,
5. $x \leqslant y$,
6. $x \neq y$

For an EFS to encode the relation $S(x) = y$, we mean that for some set variable P, the EFS proves Px, y if and only if $S(x) = y$.

Moreover, given a polynomial p with positive integer coefficients (and n variables), we can represent the relation

$$y = p(x_1, \ldots, x_n)$$

EFS-generated sets

Definition

A set S (of words in some finite alphabet) is called *EFS-generated* if there is an EFS that proves Sx if and only if $x \in S$.

Proposition

If S and T are EFS-generated sets, then each of $S \cup T$, $S \cap T$ and $S \times T$ are EFS-generated.

Corollary

Any Boolean combination of equality between polynomials is EFS-generated.

In other words, quantifier-free formulas are EFS-generated.

Projections

Definition

If $W(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$ is a property of $(k + \ell)$-tuples, then the property $\exists x_1 \ldots \exists x_k W(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$ is an *existential quantification* of the property W, and the set

$$\{\langle y_1, \ldots, y_\ell \rangle : \exists x_1 \ldots \exists x_k W(x_1, \ldots, x_k, y_1, \ldots, y_\ell)\}$$

is the corresponding *projection* of the set

$$\{\langle x_1, \ldots, x_k, y_1, \ldots, y_\ell \rangle : W(x_1, \ldots, x_k, y_1, \ldots, y_\ell)\}$$

Proposition

If W is an EFS-generated set of $(k + \ell)$-tuples, then any projection of W is EFS-generated.

It follows that if $R(x, \overline{y})$ is EFS-generated, then so is $(\exists x)\ R(x, \overline{y})$.

Bounded quantifiers

Recall the bounded quantifiers, $(\exists x < y)$ and $(\forall x < y)$.

Bounded quantifiers

Recall the bounded quantifiers, $(\exists x < y)$ and $(\forall x < y)$.

The last part we need to prove is that bounded quantification of an EFS-generated relation is also EFS-generated.

Bounded quantifiers

Recall the bounded quantifiers, $(\exists x < y)$ and $(\forall x < y)$.

The last part we need to prove is that bounded quantification of an EFS-generated relation is also EFS-generated.

Bounded existential formulas are no problem: $(\exists y < z)\ R(y, \overline{x})$ is equivalent to

$$(\exists y)\ y < z \wedge R(y, \overline{x}).$$

We have seen how to represent existential quantifiers and Boolean combinations.

Bounded quantifiers

Bounded universal quantification is a little more fiddly. To represent $\varphi(z, \overline{x}) = (\forall y < z)\ R(y, \overline{x})$, note that

- $\varphi(1, \overline{x})$ is vacuously true,
- $[w = S(z) \wedge \varphi(z, \overline{x}) \wedge R(z, \overline{x})] \Rightarrow \varphi(w, \overline{x})$

Bounded quantifiers

Bounded universal quantification is a little more fiddly. To represent $\varphi(z, \overline{x}) = (\forall y < z)\ R(y, \overline{x})$, note that

- $\varphi(1, \overline{x})$ is vacuously true,
- $[w = S(z) \wedge \varphi(z, \overline{x}) \wedge R(z, \overline{x})] \Rightarrow \varphi(w, \overline{x})$

Thus, given an EFS generating R, we can introduce the axioms

$$\begin{array}{l} \varphi 1, \overline{x} \\ w = S(z) \Rightarrow \varphi z, \overline{x} \Rightarrow Rz, \overline{x} \Rightarrow \varphi w, \overline{x} \end{array}$$

Bounded quantifiers

Bounded universal quantification is a little more fiddly. To represent $\varphi(z, \overline{x}) = (\forall y < z)\ R(y, \overline{x})$, note that

- $\varphi(1, \overline{x})$ is vacuously true,
- $[w = S(z) \wedge \varphi(z, \overline{x}) \wedge R(z, \overline{x})] \Rightarrow \varphi(w, \overline{x})$

Thus, given an EFS generating R, we can introduce the axioms

$$
\begin{aligned}
&\varphi 1, \overline{x} \\
&w = S(z) \Rightarrow \varphi z, \overline{x} \Rightarrow Rz, \overline{x} \Rightarrow \varphi w, \overline{x}
\end{aligned}
$$

Corollary

All Σ_1 relations are EFS-generated.

Hilbert's tenth problem

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

Hilbert's tenth problem

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

Equivalently,

Is there an algorithm to solve the general problem: given two polynomials p and q with positive integer coefficients, is there a positive integer solution to $p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$?

Hilbert's tenth problem

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

Equivalently,

Is there an algorithm to solve the general problem: given two polynomials p and q with positive integer coefficients, is there a positive integer solution to $p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$?

This is plainly seen to be a Σ_1 problem:

$$(\exists x_1) \cdots (\exists x_n)\ p(x) = q(x)$$

Hilbert's tenth problem

Definition

A set S of tuples of natural numbers is *Diophantine* if it can be defined by

$$\overline{n} \in S \iff \exists x_1 \exists x_2 \ldots \exists x_k \; P(\overline{n}, \bar(x)) = 0,$$

for some Diophantine polynomial P.

The formula defining a Diophantine set is clearly Σ_1.

Hilbert's tenth problem

Definition

A set S of tuples of natural numbers is *Diophantine* if it can be defined by

$$\overline{n} \in S \iff \exists x_1 \exists x_2 \ldots \exists x_k \; P(\overline{n}, \overline(x)) = 0,$$

for some Diophantine polynomial P.

The formula defining a Diophantine set is clearly Σ_1.

Example

Consider the quadratic $ax^2 - bx + c$. Let

$$S = \{(a, b, c) \in \mathbb{N}^3 \mid (\exists x \in \mathbb{N})\; ax^2 - bx + c = 0\}$$

Then S is Diophantine, and, for example, $(1, 2, 1) \in S$ but $(1, 4, 1) \notin S$.

Hilbert's tenth problem

Remarkably, the converse is also true.

Theorem (Matiyasevich, Robinson, Davis, Putnam)

Every computably enumerable set is Diophantine. (Click here)

Hilbert's tenth problem

Remarkably, the converse is also true.

Theorem (Matiyasevich, Robinson, Davis, Putnam)

Every computably enumerable set is Diophantine. (Click here)

We have seen that $\Sigma_1 \implies$ computably enumerable; hence every Σ_1 formula is *equivalent* to a Diophantine equation.

Hilbert's tenth problem

Remarkably, the converse is also true.

Theorem (Matiyasevich, Robinson, Davis, Putnam)

Every computably enumerable set is Diophantine. (Click here)

We have seen that $\Sigma_1 \implies$ computably enumerable; hence every Σ_1 formula is *equivalent* to a Diophantine equation.

Corollary

If φ is Σ_1, then φ is equivalent to $(\exists x_1)(\exists x_2)\ldots(\exists x_n)\,\psi$, for some quantifier-free ψ.

Hilbert's tenth problem

Remarkably, the converse is also true.

Theorem (Matiyasevich, Robinson, Davis, Putnam)

Every computably enumerable set is Diophantine. (Click here)

We have seen that $\Sigma_1 \implies$ computably enumerable; hence every Σ_1 formula is *equivalent* to a Diophantine equation.

Corollary

If φ is Σ_1, then φ is equivalent to $(\exists x_1)(\exists x_2)\ldots(\exists x_n)\,\psi$, for some quantifier-free ψ.

In particular, primality of x can be expressed *existentially*.

DIOPHANTINE REPRESENTATION OF THE SET OF PRIME NUMBERS

JAMES P. JONES, DAIHACHIRO SATO, HIDEO WADA AND DOUGLAS WIENS

1. Introduction. Martin Davis, Yuri Matijasevič, Hilary Putnam and Julia Robinson [4] [8] have proven that every recursively enumerable set is Diophantine, and hence that the set of prime numbers is Diophantine. From this, and work of Putnam [12], it follows that the set of prime numbers is representable by a polynomial formula. In this article such a prime representing polynomial will be exhibited in explicit form. We prove (in Section 2)

THEOREM 1. *The set of prime numbers is identical with the set of positive values taken on by the polynomial*

$$(1)\qquad (k+2)\{1-[wz+h+j-q]^2-[(gk+2g+k+1)\cdot(h+j)+h-z]^2-[2n+p+q+z-e]^2$$
$$-[16(k+1)^3\cdot(k+2)\cdot(n+1)^2+1-f^2]^2-[e^3\cdot(e+2)(a+1)^2+1-o^2]^2-[(a^2-1)y^2+1-x^2]^2$$
$$-[16r^2y^4(a^2-1)+1-u^2]^2-[((a+u^2(u^2-a))^2-1)\cdot(n+4dy)^2+1-(x+cu)^2]^2-[n+l+v-y]^2$$
$$-[(a^2-1)l^2+1-m^2]^2-[ai+k+1-l-i]^2-[p+l(a-n-1)+b(2an+2a-n^2-2n-2)-m]^2$$
$$-[q+y(a-p-1)+s(2ap+2a-p^2-2p-2)-x]^2-[z+pl(a-p)+t(2ap-p^2-1)-pm]^2\}$$

as the variables range over the nonnegative integers.

(1) is a polynomial of degree 25 in 26 variables, $a, b, c, \ldots, z$. When nonnegative integers are substituted for these variables, the positive values of (1) coincide exactly with the set of all prime numbers 2,3,5,.... The polynomial (1) also takes on negative values, e.g., -76.