Theory of Computation 12
Undecidable Sets

Frank Stephan

Department of Computer Science

Department of Mathematics

National University of Singapore

fstephan@comp.nus.edu.sg

Repetition 1

Theorem

A function is primitive recursive iff it can be computed by a register program where the only type of goto-commands which can go backwards are For-Loops, where one cannot go into or out of a For-Loop and once the For-Loop is started, its boundaries cannot be modified and the loop-variable can only be updated by the commands of the loop itself.

Remark

One can replace the Goto-commands completely by allowing only For-Loops, If-Then-Else statements and Switch-statements which are properly nested.

For full generality of Partial-Recursive functions, one would then also need While-Loops in addition to the For-Loops.

Repetition 2

Theorem 11.3

For every partial-recursive function $\mathbf{f}$ there is a primitive recursive function $\mathbf{g}$ and a register machine $\mathbf{M}$ such that for all $\mathbf{t}$,

> If $\mathbf{f(x_1, \ldots, x_n)}$ is computed by $\mathbf{M}$ within $\mathbf{t}$ steps
> Then $\mathbf{g(x_1, \ldots, x_n, t) = f(x_1, \ldots, x_n) + 1}$
> Else $\mathbf{g(x_1, \ldots, x_n, t) = 0}$.

In short words, $\mathbf{g}$ simulates the program $\mathbf{M}$ of $\mathbf{f}$ for $\mathbf{t}$ steps and if an output $\mathbf{y}$ comes then $\mathbf{g}$ outputs $\mathbf{y + 1}$ else $\mathbf{g}$ outputs $\mathbf{0}$.

$$\mathbf{f(x_1, \ldots, x_n) = g(x_1, \ldots, x_n, \min\{t : g(x_1, \ldots, x_n, t) > 0\}) - 1}.$$

Repetition 3

Theorem 11.4

The following notions are equivalent for a set $\mathbf{A} \subseteq \mathbb{N}$:

(a) $\mathbf{A}$ is the range of a partial recursive function;

(b) $\mathbf{A}$ is empty or $\mathbf{A}$ is the range of a total recursive function;

(c) $\mathbf{A}$ is empty or $\mathbf{A}$ is the range of a primitive recursive function;

(d) $\mathbf{A}$ is the set of inputs on which some register machine terminates;

(e) $\mathbf{A}$ is the domain of a partial recursive function;

(f) There is a two-place recursive function $\mathbf{g}$ such that $\mathbf{A} = \{\mathbf{x} : \exists \mathbf{y}\, [\mathbf{g}(\mathbf{x}, \mathbf{y}) > \mathbf{0}]\}$.

Definition 11.5

The set $\mathbf{A}$ is recursively enumerable iff it satisfies any of the above equivalent properties.

Repetition 4

A set $\mathbf{L}$ is called decidable or recursive iff there is a recursive function $\mathbf{f}$ such that, for all $\mathbf{x}$, if $\mathbf{x} \in \mathbf{L}$ then $\mathbf{f}(\mathbf{x}) = \mathbf{1}$ else $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. One says that the function $\mathbf{f}$ decides the membership in $\mathbf{L}$.

A set $\mathbf{L}$ is called undecidable or nonrecursive iff there is no such recursive function $\mathbf{f}$ deciding the membership in $\mathbf{L}$.

Observation
Every recursive set is recursively enumerable.

Theorem (given as Exercise)
A set is recursive if both the set and its complement are recursively enumerable.

Repetition 5

Definition [Turing 1936]
Let $\mathbf{e}, \mathbf{x} \mapsto \varphi_{\mathbf{e}}(\mathbf{x})$ be a universal partial recursive function covering all one-variable partial recursive functions. Then the set $\{(\mathbf{e}, \mathbf{x}) : \varphi_{\mathbf{e}}(\mathbf{x})$ is defined$\}$ is called the general halting problem and $\{\mathbf{e} : \varphi_{\mathbf{e}}(\mathbf{e})$ is defined $\}$ is called the diagonal halting problem.

The name stems from the fact that $\varphi_{\mathbf{e}}(\mathbf{x})$ is defined iff the $\mathbf{e}$-th register machine with input $\mathbf{x}$ halts and produces some output.

Theorem [Turing 1936]
Both the diagonal halting problem and the general halting problem are recursively enumerable and undecidable.

Hilbert's Tenth Problem

Polynomials $\mathbf{P}(\mathbb{N})$ and $\mathbf{P}(\mathbb{Z})$

A function $\mathbf{f(x, y_1, \ldots, y_n)}$ is in $\mathbf{P}(\mathbb{Z})$ if it is a sum of products of integer constants and variables $\mathbf{x, y_1, \ldots, y_n}$. This function is in $\mathbf{P}(\mathbb{N})$ iff all the integer constants occurring are in $\mathbb{N}$.

Diophantine Sets

A set $\mathbf{A} \subseteq \mathbb{N}$ is Diophantine iff there is $\mathbf{f} \in \mathbf{P}(\mathbb{Z})$ such that $\mathbf{x} \in \mathbf{A} \Leftrightarrow \exists \mathbf{y_1, \ldots, y_n} \in \mathbb{N}\, [\mathbf{f(x, y_1, \ldots, y_n) = 0}]$.

Hilbert's Tenth Problem [1900]

Construct an algorithm which decides the membership of a given Diophantine set.

Alternative Characterisations

The following conditions are equivalent for a set $\mathbf{A} \subseteq \mathbb{N}$:

- $\mathbf{A}$ is Diophantine;
- There is $\mathbf{f} \in \mathbf{P}(\mathbb{Z})$ such that $\mathbf{x} \in \mathbf{A} \Leftrightarrow \exists \mathbf{y_1}, \ldots, \mathbf{y_n} \in \mathbb{N}\, [\mathbf{f}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{0}]$;
- There are $\mathbf{f}, \mathbf{g} \in \mathbf{P}(\mathbb{N})$ such that $\mathbf{x} \in \mathbf{A} \Leftrightarrow \exists \mathbf{y_1}, \ldots, \mathbf{y_n} \in \mathbb{N}$ $[\mathbf{f}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{g}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n})]$;
- There is $\mathbf{f} \in \mathbf{P}(\mathbb{Z})$ such that $\mathbf{x} \in \mathbf{A} \Leftrightarrow \exists \mathbf{y_1}, \ldots, \mathbf{y_n} \in \mathbb{Z}\, [\mathbf{f}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{0}]$;
- There is $\mathbf{f} \in \mathbf{P}(\mathbb{Z})$ such that $\mathbf{x} \in \mathbf{A} \Leftrightarrow \exists \mathbf{y_1}, \ldots, \mathbf{y_n} \in \mathbb{Z}\, [\mathbf{f}(\mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{x}]$.

Examples

Lagrange's Four-Square Theorem: Every natural number is the sum of four squares of natural numbers.

Composite Numbers

$\mathbf{x}$ is composite iff $\exists \mathbf{y_1}, \ldots, \mathbf{y_8} \in \mathbb{Z}$ $[\mathbf{x} = (\mathbf{2} + \mathbf{y_1^2} + \mathbf{y_2^2} + \mathbf{y_3^2} + \mathbf{y_4^2}) \cdot (\mathbf{2} + \mathbf{y_5^2} + \mathbf{y_6^2} + \mathbf{y_7^2} + \mathbf{y_8^2})]$ iff $\exists \mathbf{y_1}, \mathbf{y_2} \in \mathbb{N}\, [\mathbf{x} = (\mathbf{2} + \mathbf{y_1}) \cdot (\mathbf{2} + \mathbf{y_2})]$.

Square Numbers

$\mathbf{x}$ is a square iff $\exists \mathbf{y_1} \in \mathbb{N}\, [\mathbf{x} = \mathbf{y_1^2}]$.

Non-Square Numbers

$\mathbf{x}$ is not a square iff $\exists \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3} \in \mathbb{N}$ $[\mathbf{x} = \mathbf{y_1^2} + \mathbf{1} + \mathbf{y_2}$ and $\mathbf{x} + \mathbf{y_3} = \mathbf{y_1^2} + \mathbf{2y_1}]$ iff $\exists \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3} \in \mathbb{N}$ $[(\mathbf{y_1^2} + \mathbf{1} + \mathbf{y_2} - \mathbf{x})^2 + (\mathbf{x} + \mathbf{y_3} - \mathbf{y_1^2} - \mathbf{2y_1})^2 = \mathbf{0}]$.

Exercises

Quiz

Which numbers are in the following Diophantine sets:

(a) $\{\mathbf{x} \in \mathbb{N} : \exists \mathbf{y} \in \mathbb{N}\,[\mathbf{x} = \mathbf{4} \cdot \mathbf{y} + \mathbf{2}]\}$;

(b) $\{\mathbf{x} \in \mathbb{N} : \exists \mathbf{y} \in \mathbb{N}\,[\mathbf{x}^{\mathbf{16}} = \mathbf{17} \cdot \mathbf{y} + \mathbf{1}]\}$?

Exercise 12.4

Show that the set of odd multiples of $\mathbf{97}$ is Diophantine.

Exercise 12.5

Show that the set of all numbers which are multiples of $\mathbf{5}$ but not multiples of $\mathbf{7}$ is Diophantine.

Exercise 12.6

The set $\{\mathbf{x} \in \mathbb{N} : \exists \mathbf{y_1}, \mathbf{y_2} \in \mathbb{N}\,[((\mathbf{2y_1} + \mathbf{3}) \cdot \mathbf{y_2}) - \mathbf{x} = \mathbf{0}]\}$ is Diophantine. Give a verbal description for this set.

Diophantine $\Rightarrow$ Rec. Enumerable

Proposition 12.7
Every Diophantine set is recursively enumerable.

Proof. Let $\mathbf{A}$ be Diophantine and non-empty and let $\mathbf{a} \in \mathbf{A}$. There is a polynomial $\mathbf{p}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n})$ in $\mathbf{P}(\mathbb{Z})$ such that

$$\mathbf{x} \in \mathbf{A} \Leftrightarrow \exists \mathbf{y_1}, \ldots, \mathbf{y_n} \in \mathbb{N}\, [\mathbf{p}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{0}].$$

There is a register machine computing a function $\mathbf{f}$ with input $\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}$ such that if $\mathbf{p}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{0}$ then $\mathbf{f}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{x}$ else $\mathbf{f}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) = \mathbf{a}$. $\mathbf{A}$ is the range of a recursive function and recursively enumerable.

Closure Properties

Proposition 12.8

If $\mathbf{A}, \mathbf{B}$ are Diophantine sets so are $\mathbf{A} \cup \mathbf{B}$ and $\mathbf{A} \cap \mathbf{B}$.

Proof

Let $\mathbf{p}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n})$ and $\mathbf{q}(\mathbf{x}, \mathbf{z_1}, \ldots, \mathbf{z_m})$ witness that $\mathbf{A}$ and $\mathbf{B}$ are Diophantine, respectively, that is, $\mathbf{x}$ is in $\mathbf{A}$ or $\mathbf{B}$ when the corresponding polynomial can be made $\mathbf{0}$ by choosing the other variables accordingly.

Now $\mathbf{x}$ is in $\mathbf{A} \cup \mathbf{B}$ iff $\mathbf{p}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}) \cdot \mathbf{q}(\mathbf{x}, \mathbf{z_1}, \ldots, \mathbf{z_m}) = \mathbf{0}$ for some $\mathbf{y_1}, \ldots, \mathbf{y_n}, \mathbf{z_1}, \ldots, \mathbf{z_m} \in \mathbb{N}$ and $\mathbf{x} \in \mathbf{A} \cap \mathbf{B}$ iff $(\mathbf{p}(\mathbf{x}, \mathbf{y_1}, \ldots, \mathbf{y_n}))^{\mathbf{2}} + (\mathbf{q}(\mathbf{x}, \mathbf{z_1}, \ldots, \mathbf{z_m}))^{\mathbf{2}} = \mathbf{0}$ for some $\mathbf{y_1}, \ldots, \mathbf{y_n}, \mathbf{z_1}, \ldots, \mathbf{z_m} \in \mathbb{N}$. Thus $\mathbf{A} \cup \mathbf{B}$ and $\mathbf{A} \cap \mathbf{B}$ are Diophantine.

Exercise 12.9

Given a Diophantine $\mathbf{A}$, show that the set $\mathbf{B} = \{\mathbf{x} \in \mathbb{N} : \exists \mathbf{x}' \in \mathbb{N}\,[(\mathbf{x} + \mathbf{x}')^{\mathbf{2}} + \mathbf{x} \in \mathbf{A}]\}$ is also Diophantine.

Hilbert’s Tenth Problem

Hilbert’s List of 23 Problems [1900]
No 10: Provide an algorithm to decide membership in Diophantine sets.

Entscheidungsproblem [1928]
Given a first-order formula in some logical language without free variables, check whether the formula is true in all structures for this language.

Theorem [Church 1936; Turing 1936]
The Entscheidungsproblem is undecidable.

Theorem [Matiyasevich 1970]
A set of natural numbers is Diophantine iff it is recursively enumerable. Thus the algorithm for which Hilbert asked in his Tenth Problem does not exist.

Open Question
Is the version of Diophantine sets for $\mathbb{Q}$ undecidable?

Arithmetics

Definition 12.11
A set $\mathbf{A} \subseteq \mathbb{N}$ is called arithmetic iff there is a formula using existential $(\exists)$ and universal $(\forall)$ quantifiers over variables such that all variables except for $\mathbf{x}$ are quantified and that the predicate behind the quantifiers only uses Boolean combinations of polynomials from $\mathbf{P}(\mathbb{N})$ compared by $<$ and $=$ in order to evaluate the formula.

Example 12.12
The set $\mathbf{P}$ of all prime numbers is defined by

$$\mathbf{x} \in \mathbf{P} \Leftrightarrow \forall \mathbf{y}, \mathbf{z}\, [\mathbf{x} > \mathbf{1} \text{ and } (\mathbf{y}+\mathbf{2}) \cdot (\mathbf{z}+\mathbf{2}) \neq \mathbf{x}]$$

and also every Diophantine set is arithmetic. Arithmetic sets are closed under union, difference and complement.

Powers of 2 and of Primes

Example 12.12

The set $\mathbf{T}$ of all powers of $\mathbf{2}$ is defined by

$$\mathbf{x} \in \mathbf{T} \Leftrightarrow \forall \mathbf{y}, \mathbf{y}' \, \exists \mathbf{z} \, [\mathbf{x} > \mathbf{0} \text{ and } \\ (\mathbf{x} = \mathbf{y} \cdot \mathbf{y}' \Rightarrow (\mathbf{y} = \mathbf{1} \text{ or } \mathbf{y} = \mathbf{2} \cdot \mathbf{z}))]$$

and, in general, the set $\mathbf{E}$ of all prime powers is defined by

$$(\mathbf{p}, \mathbf{x}) \in \mathbf{E} \Leftrightarrow \forall \mathbf{y}, \mathbf{y}' \, \exists \mathbf{z} \, [\mathbf{p} > \mathbf{1} \text{ and } \mathbf{x} \geq \mathbf{p} \text{ and } \\ (\mathbf{x} = \mathbf{y} \cdot \mathbf{y}' \Rightarrow (\mathbf{y} = \mathbf{1} \text{ or } \mathbf{y} = \mathbf{p} \cdot \mathbf{z}))]$$

which says that $(\mathbf{p}, \mathbf{x}) \in \mathbf{E}$ iff $\mathbf{p}$ is a prime number and $\mathbf{x}$ is a non-zero power of $\mathbf{p}$. In the last equations, $\mathbf{E}$ is a subset of $\mathbb{N} \times \mathbb{N}$ rather than $\mathbb{N}$ itself.

Configurations of RM

Configuration at step t

Tuple consisting of line number $\mathbf{LN}$ and register contents $\mathbf{R_1, \ldots, R_n}$ of the $\mathbf{n}$ registers at time $\mathbf{t}$; it is the register content before doing the instructions of line $\mathbf{LN}$.

Update Set U

$\mathbf{U}$ contains tuples of form updates of tuples of the form

$$\mathbf{(LN, R_1, \ldots, R_n, LN', R'_1, \ldots, R'_n, p)}$$

where $\mathbf{p}$ is upper bound on all other values and the register machine, in the case that it is in line $\mathbf{LN}$ and contains $\mathbf{R_1, \ldots, R_n}$ in the registers, it updates in one step to line $\mathbf{LN'}$ and has the register contents $\mathbf{R'_1, \ldots, R'_n}$.

Sample Program

For giving a sample value of $\mathbf{U}$, consider the following, a bit condensed program – otherwise the formula gets too big for slides.

```
Line 1: Function Sum$(\mathbf{R_1})$; $\mathbf{R_2 = 0}$; $\mathbf{R_3 = 0}$;
Line 2: $\mathbf{R_2 = R_2 + R_3}$; $\mathbf{R_3 = R_3 + 1}$;
Line 3: If $\mathbf{R_3 \leq R_1}$ Then Goto Line 2;
Line 4: Return$(\mathbf{R_2})$;
```

This program computes $\mathbf{0 + 1 + \ldots + R_1}$.

Plan

Formalise one-step update in arithmetics;
Formalise run of register machine in arithmetics;
Formalise $\mathbf{f(x) = y}$ in arithmetics.

Arithmetic Formula for $\mathbf{U}$

$(\mathbf{LN}, \mathbf{R_1}, \mathbf{R_2}, \mathbf{R_3}, \mathbf{LN'}, \mathbf{R'_1}, \mathbf{R'_2}, \mathbf{R'_3}, \mathbf{p})$ is in $\mathbf{U}$ iff
$\mathbf{LN} < \mathbf{p}$ and $\mathbf{R_1} < \mathbf{p}$ and $\mathbf{R_2} < \mathbf{p}$ and $\mathbf{R_3} < \mathbf{p}$ and $\mathbf{LN'} < \mathbf{p}$ and $\mathbf{R'_1} < \mathbf{p}$ and $\mathbf{R'_2} < \mathbf{p}$ and $\mathbf{R'_3} < \mathbf{p}$ and
$[(\mathbf{LN} = \mathbf{1}$ and $\mathbf{LN'} = \mathbf{2}$ and $\mathbf{R'_1} = \mathbf{R_1}$ and $\mathbf{R'_2} = \mathbf{0}$ and
$\mathbf{R'_3} = \mathbf{0})$ or
$(\mathbf{LN} = \mathbf{2}$ and $\mathbf{LN'} = \mathbf{3}$ and $\mathbf{R'_1} = \mathbf{R_1}$ and $\mathbf{R'_2} = \mathbf{R_2} + \mathbf{R_3}$ and
$\mathbf{R'_3} = \mathbf{R_3} + \mathbf{1})$ or
$(\mathbf{LN} = \mathbf{3}$ and $\mathbf{LN'} = \mathbf{2}$ and $\mathbf{R'_1} = \mathbf{R'_3}$ and $\mathbf{R'_2} = \mathbf{R_2}$ and
$\mathbf{R'_3} = \mathbf{R_3}$ and $\mathbf{R_3} \leq \mathbf{R_1})$ or
$(\mathbf{LN} = \mathbf{3}$ and $\mathbf{LN'} = \mathbf{4}$ and $\mathbf{R'_1} = \mathbf{R'_3}$ and $\mathbf{R'_2} = \mathbf{R_2}$ and
$\mathbf{R'_3} = \mathbf{R_3}$ and $\mathbf{R_3} > \mathbf{R_1})]$.

Long programs and many registers give long formulas. For each register machine, one can define the corresponding $\mathbf{U}$.

Run of Register Machine

Run with $\mathbf{p = 10}$

Sequence of register contents and line numbers per step; contents always prior to action of the current line.

```
LN: 1 2 3 2 3 2 3 2 3 4
R1: 3 3 3 3 3 3 3 3 3 3
R2: 0 0 0 0 1 1 3 3 6 6
R3: 0 0 1 1 2 2 3 3 4 4
```

Code these values as single numbers: For $\mathbf{p = 10}$, the digit for $\mathbf{10^t}$ contains the values of $\mathbf{LN}$ and $\mathbf{R_1, R_2, R_3}$ prior to step $\mathbf{t}$. So $\mathbf{LN = 4323232321}$, $\mathbf{R_1 = 3333333333}$, $\mathbf{R_2 = 6633110000}$ and $\mathbf{R_3 = 4433221100}$. In general, one uses prime $\mathbf{p}$ greater than all values of $\mathbf{LN, R_1, R_2, R_3}$ occurring throughout the run.

Formula for “Sum(x) Computes y”

$\mathrm{Sum}(x) = y$ iff there are p, q and a computation (LN, R_1, R_2, R_3) coded up using p, q such that

1. $(p, q) \in E$ and
2. $\exists r_{LN}, r_1\ [R_1 = r_1 \cdot p + x$ and $LN = r_{LN} \cdot p + 1$ and $p > x + 1]$ and
3. $\exists r_2, r_{LN} < q\, [R_2 = q \cdot y + r_2$ and $LN = q \cdot 4 + r_{LN}$ and $p > y + 4]$ and
4. $\forall p', p'' \leq q\ \exists r_{LN}, r_1, r_2, r_3, r'_{LN}, r'_1, r'_2, r'_3, r''_{LN}, r''_1, r''_2, r''_3, r'''_{LN}, r'''_1, r'''_2, r'''_3 [$ if $p' \cdot p'' = q$ and $p' < q$ then $r_{LN} < p'$ and $LN = r_{LN} + p' \cdot r'_{LN} + p' \cdot p \cdot r''_{LN} + p' \cdot p^2 \cdot r'''_{LN}$ and $r_1 < p'$ and $R_1 = r_1 + p' \cdot r'_1 + p' \cdot p \cdot r''_1 + p' \cdot p^2 \cdot r'''_1$ and $r_2 < p'$ and $R_2 = r_2 + p' \cdot r'_2 + p' \cdot p \cdot r''_2 + p' \cdot p^2 \cdot r'''_2$ and $r_3 < p'$ and $R_3 = r_3 + p' \cdot r'_3 + p' \cdot p \cdot r''_3 + p' \cdot p^2 \cdot r'''_3$ and $(r'_{LN}, r'_1, r'_2, r'_3, r''_{LN}, r''_1, r''_2, r''_3, p) \in U]$.

Formalising the Halting Problem

Thus there exists an arithmetic formula $\mathbf{R}$ with $\mathbf{Sum(x) = y \Leftrightarrow R(x, y)}$ where $\mathbf{R}$ is the formula from the last slide.

This formula is based on arithmetics of the natural numbers, $(\mathbb{N}, +, \cdot)$, and uses both types of quantifier $(\exists, \forall)$ as well as $<, =$ for comparing numbers and Boolean connectives to connect subformulas. $\mathbf{R(x, y)}$ is true iff the register machine computes from input $\mathbf{x}$ the output $\mathbf{y}$.

Furthermore, one can also define when this register machine halts on input $\mathbf{x}$ by saying that the machine halts on $\mathbf{x}$ iff $\exists \mathbf{y}\, [(\mathbf{x}, \mathbf{y}) \in \mathbf{R}]$.

This can be generalised to $(\mathbf{e}, \mathbf{x}) \in \mathbf{H}$ iff the $\mathbf{e}$-th register machine with input $\mathbf{x}$ halts; the corresponding $\mathbf{H}$ uses a translation of a universal register machine.

Undecidability of Arithmetics

Theorem 12.15 [Church 1936, Turing 1936]
There is an arithmetic set $\mathbf{L}$ which is undecidable.

An example of such an arithmetic set is the halting problem $\mathbf{H}$ or the diagonal halting problem $\mathbf{K} = \{\mathbf{e} : (\mathbf{e}, \mathbf{e}) \in \mathbf{H}\}$.

One might ask whether every arithmetic set is recursively enumerable. The next results will show that this is not the case.

Definition 12.16
A set $\mathbf{I} \subseteq \mathbb{N}$ is an index set iff for all $\mathbf{d}, \mathbf{e} \in \mathbb{N}$, if $\varphi_{\mathbf{d}} = \varphi_{\mathbf{e}}$ then either $\mathbf{d}, \mathbf{e}$ are both in $\mathbf{I}$ or $\mathbf{d}, \mathbf{e}$ are both outside $\mathbf{I}$.

Acceptable Numberings

The definition of an index set has implicit the notion of the numbering on which it is based. For getting the intended results, one has to assume that the numbering has a certain property which is called “acceptable”.

Definition 12.17: Acceptable Numbering [Gödel 1931]
A numbering $\varphi_{\mathbf{e}}$ of partial functions is a partial-recursive function $\mathbf{e}, \mathbf{x} \mapsto \varphi_{\mathbf{e}}(\mathbf{x})$. A numbering is acceptable iff for every further numbering ψ there is a recursive function $\mathbf{f}$ such that, for all $\mathbf{e}$, $\psi_{\mathbf{e}} = \varphi_{\mathbf{f}(\mathbf{e})}$.

That is, $\mathbf{f}$ translates “indices” or “programs” of ψ into “indices” or “programs” of φ which do the same.

The universal functions for register machines and for Turing machines constructed by Turing and others are actually acceptable numberings.

A Useful Proposition

Proposition 12.18
Let φ be an acceptable numbering and $\mathbf{f}$ be a partial-recursive function with $\mathbf{n+1}$ inputs. Then there is a recursive function $\mathbf{g}$ with $\mathbf{n}$ inputs such that

$$\forall \mathbf{e_1, \ldots, e_n, x}\,[\mathbf{f(e_1, \ldots, e_n, x)} = \varphi_{\mathbf{g(e_1,\ldots,e_n)}}\mathbf{(x)}].$$

Proof Idea for $\mathbf{n = 2}$: There is a recursive bijection $\mathbf{c(e_1, e_2) = (e_1 + e_2) \cdot (e_1 + e_2 + 1)/2 + e_2}$ from $\mathbb{N}^{\mathbf{2}}$ to $\mathbb{N}$.
Define ψ by

$$\forall \mathbf{e_1, e_2, x}\,[\psi_{\mathbf{c(e_1,e_2)}}\mathbf{(x)} = \mathbf{f(e_1, e_2, x)}].$$

There is a recursive $\tilde{\mathbf{g}}$ translating ψ indices into φ indices.
The recursive function $\mathbf{g(e_1, e_2)} = \tilde{\mathbf{g}}\mathbf{(c(e_1, e_2))}$ satisfies

$$\forall \mathbf{e_1, e_2, x}\,[\varphi_{\mathbf{g(e_1,e_2)}}\mathbf{(x)} = \mathbf{f(e_1, e_2, x)}].$$

Rice's Theorem

Theorem 12.19 [Rice 1953]
Let φ be an acceptable numbering and $\mathbf{I}$ be an index set (with respect to φ).

(a) The set $\mathbf{I}$ is recursive iff $\mathbf{I} = \emptyset$ or $\mathbf{I} = \mathbb{N}$.

(b) The set $\mathbf{I}$ is recursively enumerable iff there is a recursive enumeration of finite lists $(\mathbf{x_1, y_1, \ldots, x_n, y_n})$ of conditions such that every index $\mathbf{e}$ satisfies that $\mathbf{e} \in \mathbf{I}$ iff there is a list $(\mathbf{x_1, y_1, \ldots, x_n, y_n})$ in the enumeration for which $\varphi_\mathbf{e}(\mathbf{x_1}) = \mathbf{y_1}$ and $\ldots$ and $\varphi_\mathbf{e}(\mathbf{x_n}) = \mathbf{y_n}$.

Corollary 12.20
Let $\mathbf{I} = \{\mathbf{e} : \forall \mathbf{x}\, [\varphi_\mathbf{e}(\mathbf{x})$ is defined$]\}$. The set $\mathbf{I}$ of indices of total functions is arithmetic and not recursively enumerable.

Proof of Theorem 12.19 (b), Part 1

Assume that there is an enumeration $\mathbf{E}$ of conditions such that $\mathbf{e} \in \mathbf{I}$ iff there is $(\mathbf{x_1}, \mathbf{y_1}, \ldots, \mathbf{x_n}, \mathbf{y_n}) \in \mathbf{E}$ with $\varphi_\mathbf{e}(\mathbf{x_1}) = \mathbf{y_1}, \ldots, \varphi_\mathbf{e}(\mathbf{x_n}) = \mathbf{y_n}$.

If $\varphi_\mathbf{e}$ satisfies the $\mathbf{d}$-th condition in $\mathbf{E}$ then $\mathbf{f}(\mathbf{d}, \mathbf{e}) = \mathbf{e}$ else $\mathbf{f}(\mathbf{d}, \mathbf{e})$ is undefined. The function $\mathbf{f}$ is partial recursive and $\mathbf{I}$ is the range of a partial recursive function, thus $\mathbf{I}$ is recursively enumerable.

Now assume that $\mathbf{I}$ is recursively enumerable and define the following partial recursive function $\mathbf{f}$: If $\varphi_\mathbf{i}(\mathbf{x})$ gets computed at time $\mathbf{t}$ and $\varphi_\mathbf{j}(\mathbf{j})$ does not get computed within time $\mathbf{t} + \mathbf{x}$ then let $\mathbf{f}(\mathbf{i}, \mathbf{j}, \mathbf{x}) = \varphi_\mathbf{i}(\mathbf{x})$ else $\mathbf{f}(\mathbf{i}, \mathbf{j}, \mathbf{x})$ remains undefined. By Proposition 12.18 there is a recursive function $\mathbf{g}$ such that $\forall \mathbf{i}, \mathbf{j}, \mathbf{x}\, [\varphi_{\mathbf{g}(\mathbf{i}, \mathbf{j})}(\mathbf{x}) = \mathbf{f}(\mathbf{i}, \mathbf{j}, \mathbf{x})]$. Consider any $\mathbf{i} \in \mathbf{I}$.

Proof of Theorem 12.19 (b), Part 2

For all $\mathbf{j}$ with $\varphi_\mathbf{j}(\mathbf{j})$ being undefined, it holds that $\varphi_\mathbf{i} = \varphi_{\mathbf{g(i,j)}}$ and $\mathbf{g(i,j)} \in \mathbf{I}$.

The set $\{\mathbf{j} : \mathbf{g(i,j)} \in \mathbf{I}\}$ is recursively enumerable and $\{\mathbf{j} : \varphi_\mathbf{j}(\mathbf{j})$ is undefined$\}$ is not; thus there are $\mathbf{j}$ with $\varphi_\mathbf{j}(\mathbf{j})$ being defined and $\mathbf{g(i,j)} \in \mathbf{I}$. For all such $\mathbf{j}$ one can compute the $(\mathbf{x},\mathbf{y})$ with $\varphi_{\mathbf{g(i,j)}}(\mathbf{x}) = \mathbf{y}$ explicitly as the time of the computation $\varphi_{\mathbf{g(i,j)}}(\mathbf{x})$ needed plus $\mathbf{x}$ must be below the time the computation $\varphi_\mathbf{j}(\mathbf{j})$ needs. Now let $\mathbf{E}$ be a recursive enumeration of all lists obtained for such $\mathbf{i},\mathbf{j}$ with $\mathbf{i} \in \mathbf{I}$ and $\varphi_\mathbf{j}(\mathbf{j})$ being defined; $\mathbf{E}$ satisfies these conditions:

- If $\mathbf{e} \in \mathbf{I}$ then a there is a list $(\mathbf{x_1},\mathbf{y_1},\ldots,\mathbf{x_n},\mathbf{y_n}) \in \mathbf{E}$ such that $\varphi_\mathbf{e}(\mathbf{x_1}) = \mathbf{y_1},\ldots,\varphi_\mathbf{e}(\mathbf{x_n}) = \mathbf{y_n}$;
- If $(\mathbf{x_1},\mathbf{y_1},\ldots,\mathbf{x_n},\mathbf{y_n})$ appears in $\mathbf{E}$ then there is an index $\mathbf{e} \in \mathbf{I}$ such that $\varphi_\mathbf{e}$ is defined exactly at $\mathbf{x_1},\ldots,\mathbf{x_n}$ and takes the values $\mathbf{y_1},\ldots,\mathbf{y_n}$, respectively.

Proof of Theorem 12.19 (b), Part 3

Assume that φ_i extends a list $(x_1, y_1, \ldots, x_n, y_n)$ in E and define $f'(j, x)$ according the first condition found to apply:

- If $x = x_m$ then $f'(j, x) = y_m$;
- If $\varphi_j(j)$ halts and $\varphi_i(x)$ halts then $f'(j, x) = \varphi_i(x)$;
- If none of the above conditions are satisfied then $f'(j, x)$ is undefined.

The function f' is partial recursive and by Proposition 12.18 there is a recursive function g' with $\forall j, x\, [\varphi_{g'(j)}(x) = f'(j, x)]$.

If $i \in I$ then $\{j : g'(j) \in I\} = \mathbb{N}$;
If $i \notin I$ then $\{j : g'(j) \in I\} = \{j : \varphi_j(j)$ is undefined$\}$.
The latter cannot hold as $\{j : g'(j) \in I\}$ is recursively enumerable; thus $i \in I$.

So the conditions in E describe I.

Proof of Theorem 12.19 (a)

It is obvious that $\emptyset$ and $\mathbb{N}$ are recursive index sets.

So assume now that $\mathbf{I}$ is a recursive index set. Then both $\mathbf{I}$ and $\mathbb{N} - \mathbf{I}$ are recursively enumerable. One of these sets, say $\mathbf{I}$, contains an index $\mathbf{e}$ of the everywhere undefined function.

By part **(b)**, there is a recursively enumerable set $\mathbf{E}$ of conditions to describe the indices in $\mathbf{I}$. This enumeration $\mathbf{E}$ must contain the empty list, as otherwise the index $\mathbf{e}$ would not qualify to be in $\mathbf{I}$. Now every index satisfies the condition for the empty list and therefore $\mathbf{I} = \mathbb{N}$.

Thus $\emptyset$ and $\mathbb{N}$ are the only two recursive index sets.

Index Set of Total Functions

Recall that the halting problem

$$\mathbf{H} = \{(\mathbf{e},\mathbf{x}) : \varphi_{\mathbf{e}}(\mathbf{x}) \text{ is defined}\}$$

is definable in arithmetic. Thus also the set

$$\mathbf{T} = \{\mathbf{e} : \forall \mathbf{x}\, [(\mathbf{e},\mathbf{x}) \in \mathbf{H}]\}$$

of indices of all total functions is also arithmetic. If this set would be recursively enumerable then there would recursive enumeration of lists of finite conditions such that when a function satisfies one list of conditions then it is in the index set. However, for each such list there is a function with finite domain satisfying it, hence $\mathbf{T}$ would contain an index of a function with a finite domain, a contradiction. So $\mathbf{T}$ is not recursively enumerable.

Many-One Reducibility

The proof of Rice's Theorem and also the above proof have implicitly used the following observation.

Observation 12.21

If $\mathbf{A}, \mathbf{B}$ are sets and $\mathbf{B}$ is recursively enumerable and if there is a recursive function $\mathbf{g}$ with $\mathbf{x} \in \mathbf{A} \Leftrightarrow \mathbf{g}(\mathbf{x}) \in \mathbf{B}$ then $\mathbf{A}$ is also recursively enumerable. Similarly, if $\mathbf{B}$ is recursive then so is $\mathbf{A}$.

The first can be seen as if $\mathbf{B}$ is the domain of a partial recursive function $\mathbf{h}$ then $\mathbf{A}$ is the domain of the partial recursive function $\mathbf{x} \mapsto \mathbf{h}(\mathbf{g}(\mathbf{x}))$. The second follows from the fact that a set is recursive iff both the set and its complement are recursively enumerable.

Definition 12.22

A set $\mathbf{A}$ is many-one reducible to a set $\mathbf{B}$ iff there is a recursive function $\mathbf{g}$ such that, for all $\mathbf{x}$, $\mathbf{x} \in \mathbf{A} \Leftrightarrow \mathbf{g}(\mathbf{x}) \in \mathbf{B}$.

Proving Undecidability

If $\mathbf{A}$ is not recursive / recursively enumerable and $\mathbf{A}$ is many-one reducible to $\mathbf{B}$ then $\mathbf{B}$ is not recursive / recursively enumerable, respectively.

Example 12.23
The set $\mathbf{E} = \{\mathbf{e} : \forall$ even $\mathbf{x}\, [\varphi_\mathbf{e}(\mathbf{x})$ is defined$]\}$ is not recursively enumerable: Define $\mathbf{f}(\mathbf{e}, \mathbf{2x}) = \varphi_\mathbf{e}(\mathbf{x})$ and $\mathbf{f}(\mathbf{e}, \mathbf{2x+1}) = \varphi_\mathbf{e}(\mathbf{x})$. There is recursive $\mathbf{g}$ with $\forall \mathbf{e}, \mathbf{x}\, [\varphi_{\mathbf{g(e)}}(\mathbf{x}) = \mathbf{f}(\mathbf{e}, \mathbf{x})]$. Thus $\varphi_\mathbf{e}$ is total iff $\forall \mathbf{x}\, [\varphi_{\mathbf{g(e)}}(\mathbf{2x})$ is defined$]$. So $\mathbf{g}$ is many-one reduction from indices of total functions to $\mathbf{E}$. Thus $\mathbf{E}$ is not recursively enumerable.

Example 12.24
The set $\mathbf{F} = \{\mathbf{e} : \varphi_\mathbf{e}$ is somewhere defined$\}$ is not recursive. There is a partial recursive $\mathbf{f}(\mathbf{e}, \mathbf{x})$ with $\mathbf{f}(\mathbf{e}, \mathbf{x}) = \varphi_\mathbf{e}(\mathbf{e})$ and a recursive function $\mathbf{g}$ with $\varphi_{\mathbf{g(e)}}(\mathbf{x}) = \mathbf{f}(\mathbf{e}, \mathbf{x}) = \varphi_\mathbf{e}(\mathbf{e})$. Now $\mathbf{e} \in \mathbf{K}$ iff $\mathbf{g}(\mathbf{e}) \in \mathbf{F}$ and thus $\mathbf{F}$ is not recursive.

Diagonal Halting Problem

Theorem 12.25
Every recursively enumerable set is many-one reducible to the diagonal halting problem $\mathbf{K} = \{\mathbf{e} : \varphi_{\mathbf{e}}(\mathbf{e})$ is defined$\}$.

Proof. Assume that $\mathbf{A}$ is recursively enumerable. $\mathbf{A}$ is the domain of a partial recursive function $\tilde{\mathbf{f}}$. There is also a partial recursive $\mathbf{f}$ with $\mathbf{f}(\mathbf{e},\mathbf{x}) = \tilde{\mathbf{f}}(\mathbf{e})$ for all $\mathbf{e},\mathbf{x}$. By Proposition 12.18 there is a recursive $\mathbf{g}$ satisfying

$$\forall \mathbf{e},\mathbf{x}\,[\varphi_{\mathbf{g}(\mathbf{e})}(\mathbf{x}) = \mathbf{f}(\mathbf{e},\mathbf{x})].$$

If $\mathbf{e} \in \mathbf{A}$ then $\varphi_{\mathbf{g}(\mathbf{e})}$ is total and $\varphi_{\mathbf{g}(\mathbf{e})}(\mathbf{g}(\mathbf{e}))$ is defined and $\mathbf{g}(\mathbf{e}) \in \mathbf{K}$;
if $\mathbf{e} \notin \mathbf{A}$ then $\varphi_{\mathbf{g}(\mathbf{e})}$ is nowhere defined and $\varphi_{\mathbf{g}(\mathbf{e})}(\mathbf{g}(\mathbf{e}))$ is undefined and $\mathbf{g}(\mathbf{e}) \notin \mathbf{K}$.
Thus $\mathbf{g}$ is a many-one reduction from $\mathbf{A}$ to $\mathbf{K}$.

Exercises

Exercise 12.26
Show that the set $\mathbf{F} = \{\mathbf{e} : \varphi_\mathbf{e}$ is defined on at least one $\mathbf{x}\}$ is many-one reducible to the set $\{\mathbf{e} : \varphi_\mathbf{e}(\mathbf{x})$ is defined for exactly one input $\mathbf{x}\}$.

Determine for the each of the following sets whether it is recursive, recursively enumerable and non-recursive or even not recursively enumerable:

Exercise 12.27: $\mathbf{A} = \{\mathbf{e} : \forall \mathbf{x}\, [\varphi_\mathbf{e}(\mathbf{x})$ is defined iff $\varphi_\mathbf{x}(\mathbf{x}+\mathbf{1})$ is undefined$]\}$;
Exercise 12.28: $\mathbf{B} = \{\mathbf{e} :$ There are at least five numbers $\mathbf{x}$ where $\varphi_\mathbf{e}(\mathbf{x})$ is defined$\}$;
Exercise 12.29: $\mathbf{C} = \{\mathbf{e} :$ There are infinitely many $\mathbf{x}$ where $\varphi_\mathbf{e}(\mathbf{x})$ is defined$\}$.

Exercises

Exercise 12.30

Assume that φ is an acceptable numbering. Now define ψ such that

$$\psi_{(\mathbf{d}+\mathbf{e})\cdot(\mathbf{d}+\mathbf{e}+1)/2+\mathbf{e}}(\mathbf{x}) = \begin{cases} \text{undefined} & \text{if } \mathbf{d} = 0 \text{ and } \mathbf{x} = 0; \\ \mathbf{d}-1 & \text{if } \mathbf{d} > 0 \text{ and } \mathbf{x} = 0; \\ \varphi_{\mathbf{e}}(\mathbf{x}) & \text{if } \mathbf{x} > 0. \end{cases}$$

Is the numbering ψ enumerating all partial recursive functions? Is the numbering ψ an acceptable numbering?

Exercise 12.31

Is there a numbering ϑ with the following properties:

- The set $\{\mathbf{e} : \vartheta_{\mathbf{e}}$ is total$\}$ is recursively enumerable;
- Every partial recursive function $\varphi_{\mathbf{e}}$ is equal to some $\vartheta_{\mathbf{d}}$.

Prove your answer.