## 16.3 COMPUTABILITY

Most of the material presented here is presented in far more detail in the texts of Rogers [R], Odifreddi [O], and Soare [S].

In this section $\mathcal{N}$ denotes the set of natural numbers. All functions discussed will be from $\mathcal{N}$ to $\mathcal{N}$. If other domains are used then we assume they are coded into $\mathcal{N}$ in some computable way.

### 16.3.1 RECURSIVE FUNCTIONS AND THE CHURCH-TURING THESIS

**Definition:** A **partial function** is a function which is allowed to be undefined on some points of its domain. If $f$ is such a function then we denote that $f(x)$ is undefined by $f(x)\uparrow$, and that $f(x)$ is defined by $f(x)\downarrow$. A partial function is **total** if it is defined on all points of its domain.

**Definition:**

1. A **Turing Machine** is a 5-tuple $\langle Q, \Sigma, \delta, s, h\rangle$ such that $Q$ is a finite set of **states**, $\Sigma$ is a finite **alphabet**, $s \in Q$ is **the start state**, $h \in Q$ is **the halt state** and $\delta : Q - \{h\} \times \Sigma \rightarrow Q \times \Sigma \times \{R, L, S\}$. A Turing machine is interpreted as a program that computes a partial function. In this context $\delta(q, a) = (p, b, R)$ would mean that if the machine is in state $q$ and the alphabets symbols scanned is $a$ then go to state $p$, overwrite the $a$ with a $b$, and move right. $L$ stands for left and $S$ for stationary. (Alan Turing defined these machines.)

2. The class of **primitive recursive functions** is the smallest class of functions $\mathcal{P}$ such that the following hold.
   a) For all $i, n$ the function $f(x_1, \ldots, x_n) = x_i$ is in $\mathcal{P}$.
   b) For all $n$ the function $f(x_1, \ldots, x_n) = 0$ is in $\mathcal{P}$.
   c) For all $i, n$ the function $f(x_1, \ldots, x_n) = x_i + 1$ is in $\mathcal{P}$.
   d) If $f(x_1, \ldots, x_n), g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m)$ are in $\mathcal{P}$ then

   $h(x_1, \ldots, x_m) = f(g_1(x_1, \ldots, x_m), g_2(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$ is in $\mathcal{P}$.

   e) If $f(x_1, \ldots, x_{n+2}), g(x_1, \ldots, x_n) \in \mathcal{P}$ then the following function is in $\mathcal{P}$.

$$h(x_1, \ldots, x_{n+1}) = \begin{cases} g(x_1, \ldots, x_n) & \text{if } x_{n+1} = 0 \\ f(x_1, \ldots, x_{n+1}, h(x_1, \ldots, x_n, x_{n+1} - 1)) & \text{otherwise.} \end{cases}$$

3. The class of **general recursive functions** is the smallest class of partial functions such that the following hold.
   a) If $f$ is primitive recursive then $f \in \mathcal{G}$.
   b) If $f(x_1, \ldots, x_{n+1})$ is in $\mathcal{G}$ then the following function is in $\mathcal{G}$.

$$g(x_1, \ldots, x_{n+1}) = \begin{cases} \text{the least } y \text{ such that } f(x_1, \ldots, x_n, y) = 0 & \text{if it exists} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

4. (Terminology) A function is **partial recursive** if it if general recursive. A function that is partial recursive and also total is simply called **recursive**.

5. (Informal) An **acceptable programming system** $\varphi_1, \varphi_2, \ldots$ is a listing of the partial recursive functions such that the indices can be treated like code. For example, given $e, x$ one can determine $\varphi_e(x)$. We will assume that a statement like "Run $\varphi_e(x)$ for $t$ steps" makes sense. (It certainly does if our underlying model is a Turing machine)

**Facts:**

1. If $f$ is partial recursive then there is a computer program that computes it.

2. A partial function $f$ is computed by a Turing machine iff $f$ is general recursive.

3. There are several other models of computation that are equivalent to general recursive functions.

4. Virtually all functions encountered in ordinary mathematics are primitive recursive. Addition, multiplication, and exponentiation are primitive recursive using the following facts: $x + (y+1) = (x+y)+1$, $x(y+1) = xy + y$, $x^{y+1} = x^y x$. Other functions such as "$f(x)$ =the $x$th prime" are also primitive recursive, but this takes some work to prove.

5. As of 1994 all partial functions that are known to be computable on some real world device are general recursive.

**Definition:** The **Church-Turing Thesis** states that every intuitively computable partial function is partial recursive. By using it one can proof theorems about the partial recursive functions without dealing with the details of the model of computation.

**Examples of Partial Recursive Functions:**

1. All primitive recursive functions are partial recursive.

2. Ackerman's function is defined as follows
   $A(1, j) = 2^j$
   $A(i, 1) = A(i-1, 2)$ (for $i \geq 2$),
   $A(i, j) = A(i-1, A(i, j-1))$ (for $i, j \geq 2$).
Ackerman's function is recursive but not primitive recursive.

3. The following function is recursive

$$f(e, x, t) = \begin{cases} \varphi_e(x) & \text{if } \varphi_e(x) \downarrow; \\ 0 & \text{otherwise.} \end{cases}$$

4. The following function is partial recursive

$$g(e, x) = \begin{cases} \varphi_e(x) & \text{if } \varphi_e(x) \downarrow; \\ \uparrow & \text{otherwise.} \end{cases}$$

## 16.3.2 UNSOLVABLE PROBLEMS

**Definition:**

1. A set $A$ is **solvable** if there exists a total recursive function $f$ such that $x \in A \Rightarrow f(x) = 1$ and $x \notin A \Rightarrow f(x) = 0$. Such sets are also called **decidable** or **recursive**

2. A set $A$ is **unsolvable** if it is not solvable. Such sets are also called **undecidable** or **nonrecursive**

**Examples:** Let $\varphi_0, \varphi_1, \ldots$ be an acceptable programming system.

1. The following problems are unsolvable.
   $HALT = \{e \mid \varphi_e(e) \downarrow\} = \{e \mid (\exists t)[\varphi_e(e) \text{ halts in } \leq t \text{ steps }]\}$.

$TOT = \{e \mid \varphi_e \text{ is total}\} = \{e \mid (\forall x)(\exists t)[\varphi_e(x) \text{ halts in } t \text{ steps }]\}$.
$COF = \{e \mid \varphi \text{ halts on all but a finite number of values}\}$
$= \{e \mid (\exists x)(\forall y \geq x)(\exists t)[\varphi_e(y)\text{halts in } t \text{ steps}]\}$.

2. A set $A$ is an **index set** if, for all $x, y$, if $\varphi_x$ and $\varphi_y$ compute the exact same partial function then either $x, y \in A$ or $x, y \notin A$. Hence the question of whether or not $x \in A$ depends only on the behavior of $\varphi_x$. Rice's Theorem states that if $A$ is an index set, $A \neq \emptyset$, and $A \neq \mathcal{N}$ then $A$ is unsolvable.

3. Hilbert's 10th problem (in modern terminology) was to devise an algorithm to determine, given a polynomial $p(x_1, \ldots, x_n)$ with integer coefficients, if it has a solution in the integers. It is known that no such algorithm exists. This can be phrased as saying that the set of polynomials with integer coefficients that have an integer solution is unsolvable. This is commonly stated as "Hilberts 10th problem is unsolvable." This is considered a natural example of an unsolvable problem since the concepts used to define it are not from computability theory (as opposed to $HALT$, $INF$ and $COF$). There is a book on Hilbert's Tenth problem by Matijasevic [M] for non-logicians.

## 16.3.3 Recursively Enumerable Sets

**Definition:** The following four definitions are equivalent. Let $A \subseteq \mathcal{N}$. We abbreviate recursively enumerable by r.e.

1. $A$ is r.e. if $A$ is the range of a partial recursive function.
2. $A$ is r.e. if $A = \emptyset$ or $A$ is the range of a recursive function.
3. $A$ is r.e. if $A$ is the domain of a partial recursive function.
4. $A$ is r.e. if there exists a solvable set $B \subseteq \mathcal{N} \times \mathcal{N}$ such that $A = \{x \mid (\exists y)[\langle x, y\rangle \in B]\}$

**Facts:**

1. If $A$ and $\overline{A}$ are both r.e. then $A$ is solvable.
2. If $A$ is r.e. and not recursive then $\overline{A}$ is not r.e.
3. If $A$ is r.e. and is the range of a non-decreasing function then $A$ is recursive.

**Examples of r.e. sets:**

1. $HALT$
2. $\{e \mid \varphi_e \text{ halts on some prime }\} = \{e \mid (\exists x, t)[x \text{ is prime } \wedge \varphi_e(x) \text{ halts in } t \text{ steps }]\}$.
3. $\{e \mid \varphi_e \text{ halts on at least 100 numbers }\}$

**Examples of non-r.e. sets:**

1. $\overline{HALT}$.
2. $\{e \mid \varphi_e \text{ halts on all primes }\}$.
3. $\{e \mid \varphi_e \text{ halts on at most 100 numbers }\}$.

## 16.3.4 Reductions

**Definition:**

1. $A \leq_m B$ if there exists a recursive function $f$ such that $x \in A$ iff $f(x) \in B$. This is pronounced '$A$ is $m$**-reducible** to $B$.' The $m$ indicates that the function $f$ may be many-to-one.

2. $A \leq_T B$ if $A$ can be solved given a 'black box' for $B$. This is pronounced '$A$ is **Turing-reducible** to $B$.'

3. $A <_m B$ if $A \leq_m B$ but $B \not\leq_m A$. $A <_T B$ if $A \leq_T B$ but $B \not\leq_T A$.

4. $A \equiv_m B$ if $A \leq_m B$ and $B \leq_m A$. This is an equivalence relation. The equivalence classes are $m$**-degrees**. $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. This is an equivalence relation. The equivalence classes are **Turing degrees**.

**Facts:**

1. If $A \leq_m B$ then $A \leq_T B$. The converse does not hold.

2. If $A \leq_T B$ and $B$ is solvable then $A$ is solvable.

3. If $A \leq_m B$ and $B$ is r.e. then $A$ is r.e.

4. If $A$ is r.e. then $A \leq_m K$.

5. If $A$ is a nonempty index set, and the index for the null function (everywhere diverging) is not in $A$, then $K \leq_m A$.

6. There exists an r.e. set $A$ such that $\emptyset <_T A <_T K$. Such sets are not natural. They are usually constructed by a **priority argument**.

### 16.3.5 The Arithmetic Hierarchy

**Definitions:** Let $A$ be a set.

1. $A \in \Sigma_n$ if there exists a solvable set $B \subseteq \mathcal{N}^{n+1}$ such that

$$A = \{x \mid (\exists y_1)(\forall y_2)(\exists y_3)\cdots(Qy_n)[\langle x, y_1, \ldots, y_n\rangle \in B]\}.$$

($Q$ is $\exists$ if $n$ is odd, and is $\forall$ if $n$ is even.)

2. $A \in \Pi_n$ if $\overline{A} \in \Sigma_n$.

3. $A \in \Delta_n$ if $A \in \Sigma_n \cap \Pi_n$.

4. $A$ is in **the arithmetic hierarchy** if there exists $n$ such that $A \in \Sigma_n$.

5. $A$ is $\Sigma_n$**-complete** if $A \in \Sigma_n$ and $(\forall B \in \Sigma_n)[B \leq_m A]$. The notion of $\Pi_n$**-complete** is defined similarly.

**Facts:**

1. $\Sigma_1 \subset \Sigma_2 \subset \Sigma_3 \cdots$.

2. $\Pi_1 \subset \Pi_2 \subset \Pi_3 \cdots$.

3. $\Pi_0 = \Sigma_0$ and this is the class of all solvable sets. For all $n \geq 1$ $\Pi_n \neq \Sigma_n$.

4. If $A \in \Sigma_1 \cap \Pi_1 = \Delta_1$ then $A$ is recursive.

5. If $A \in \Sigma_2 \cap \Pi_2 = \Delta_2$ then $A \leq_T K$.

6. If $A \in \Sigma_n \cap \Pi_n = \Delta_n$ then for all $\Sigma_{n-1}$-complete set $B$, $A \leq_T B$.

7. If $A$ is $\Sigma_n$-complete and $B \leq_m A$ then $B \in \Sigma_n$. Hence $A \notin \Pi_n \cup \Sigma_{n-1}$.

8. $HALT$ is $\Sigma_1$-complete. $TOT$ is $\Pi_2$-complete. $COF$ is $\Sigma_3$-complete.

### 16.3.6 Gödel's Incompleteness Theorem

**Defintion:** An **axiom system** is a set of statements in a mathematical language. We are interested in axiom systems for first order number theory. Hence the basic language will contain $+$ and $\times$ as well as the usual logical symbols. Quantifiers range over $\mathcal{N}$. A **consistent axiom system** is a set of axioms such that it is impossible to derive a contradiction from it. A **recursive axiom system** is a set of axioms that forms a solvable set.

**Example:** The following set of axioms is a recursive axiom system.

1. $(\forall x, y)[x + (y+1) = (x+y) + 1]$.
2. $(\forall x, y)[x(y+1) = xy + x]$.
3. For all formulas $P(x)$ include the axiom

$$[P(0) \wedge (\forall x)[P(x) \Rightarrow P(x+1)]] \Rightarrow (\forall x)[P(x)].$$

**Gödel's Incompleteness Theorem:** If $AX$ is a consistent recursive axiom system then there exists a statement $S$ such that $S$ is *true of the natural numbers* but is not provable from $AX$. Any standard text in mathematical logic will have a proof or pointers to a proof.

**Facts:**

Let $TRUE$ be the set of first order statements that are true of the natural numbers. $TRUE$ is not in the arithmetic hierarchy.

## REFERENCES

[M] Y. Matijasevic, *Hilbert's Tenth Problem*, MIT press, Cambridge, 1993. (This is a book for non-logicians on both the proof and ramifications of the unsolvability of Hilbert's 10th problem.)

[O] P. Odifreddi, *Classical Recursion Theory (Volume I)*, North-Holland, Amsterdam, 1989. (Contains much on models of computation and the Church-Turing thesis.)

[R] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, McGraw Hill, New York, 1967.

[S] R. I. Soare, *Recursively Enumerable Sets and Degrees*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1987.