AN INTRODUCTION TO COMPUTABILITY THEORY

CINDY CHUNG

Abstract. This paper will give an introduction to the fundamentals of computability theory. Based on Robert Soare's textbook, The Art of Turing Computability: Theory and Applications, we examine concepts including the Halting problem, properties of Turing jumps and degrees, and Post's Theorem.[1] In the paper, we will assume the reader has a conceptual understanding of computer programs.

Contents

1. Basic Computability

Definition 1.1. An *algorithm* is a procedure capable of being carried out by a computer program. It accepts various forms of numerical inputs including numbers and finite strings of numbers.

Computability theory is a branch of mathematical logic that focuses on algorithms, formally known in this area as *computable functions*, and studies the *degree*, or level of computability that can be attributed to various sets (these concepts will be formally defined and elaborated upon below). This field was founded in the 1930s by many mathematicians including the logicians: Alan Turing, Stephen Kleene, Alonzo Church, and Emil Post. Turing first pioneered computability theory when he introduced the fundamental concept of the a-machine which is now known as the *Turing machine* (this concept will be intuitively defined below) in his 1936 paper, *On computable numbers, with an application to the Entscheidungsproblem*[7]. Entscheidungsproblem (or "decision problem" in German) refers to a challenge proposed in 1928 by German mathematician, David Hilbert, to find an algorithm to

Date: August 29, 2014.

[1]Some proofs of theorems are left out due to limitations of space and can be found in external sources.

verify the proofs of first-order logic properties [5]. In his paper, Turing showed that no such algorithm exists (this solution was independently arrived at by Church around the same time) [7].

Definition 1.2. The *Turing machine* is a conceptual, abstract mechanism in the form of an infinitely long tape and a reading head. The tape consists of an infinite array of cells, each of which contains a symbol, and the reading head can only move between the cells one at a time based on a given procedure. The program that specifies how a Turing machine operates can theoretically carry out any algorithm.

Definition 1.3. (i) A partial function, $\varphi : \mathbb{N} \to \mathbb{N}$, is *partial computable* (p.c) if there exists an algorithm which provides an output upon inputting, $n \in N$ when $\varphi(n)$ is defined (i.e it halts, $\downarrow$), but does not provide an output when $\varphi(n)$ is undefined (i.e it diverges, $\uparrow$).
(ii) Conversely, a total function, $f : \mathbb{N} \to \mathbb{N}$, is *total computable* (or just *computable*) if it is defined on all inputs in $\mathbb{N}$ (i.e $f(x) \downarrow \forall\ x$), and in turn has an algorithm that gives an output for any such input.

Note that any reasonable computer model yields the same computable functions, but the algorithms representing each function depend on the model adopted. If we fix the computer model, we can obtain a list of all possible algorithms in that model. The function computed by the e^{th} algorithm is denoted by φ_e.

Definition 1.4. A set is *computably enumerable* (c.e)[1] if it is the domain of a partial computable function, φ.
The e^{th} c.e set, W_e, is defined as:

$$W_e := dom(\varphi_e) = \{x : \varphi_e(x) \downarrow\}.$$

The concept of c.e. sets was introduced by Stephen Kleene, who focused his work on studying which functions were computable, and his PhD advisor, Alonzo Church, who made major contributions to several branches of mathematical logic such as helping with solving the Entscheidungsproblem in computability theory. Church was in fact the advisor of several prominent mathematicians, in addition to Kleene, including Turing and Post.

Definition 1.5. The *characteristic function* of a subset, A, of the natural numbers is the function χ_A with domain $\mathbb{N}$ given by:

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases}$$

Definition 1.6. A set is *computable* if it has a computable characteristic function.

2. The Halting Problem

We will now consider the halting problem which looks at whether a computer program will run on forever, $\uparrow$, or halt, $\downarrow$. By explaining that the Turing machine does not exist in his 1936 paper, Turing proved the impossibility of an algorithm that can determine whether an arbitrary computer program will halt or not, though he never mentions the term, "halting problem" in his paper [7].

[1]Note that other texts may refer to this term as recursively enumerable instead but the meanings are the same.

Definition 2.1. K is the set defined as:

$$K := \{x : \varphi_x(x) \text{ converges}\} = \{x : x \in W_x\}.$$

K is commonly referred to as the *halting set*. In 1936, Kleene, showed that the halting set is computably enumerable but not computable [5].

Theorem 2.2. *K is a c.e set.*

Proof. Let ψ be the p.c function defined as:

$$\psi(x) = \begin{cases} 1 & \text{if } \varphi_x(x) \text{ converges} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

To compute ψ, one can run $\varphi_x(x)$ until it halts (if it does), at which point ψ will stop and output 1. Since K is the domain of ψ, by definition K is a c.e set. □

Theorem 2.3. *K is not a computable set.*

Proof. Proof by contradiction: Suppose K is a computable set. Then by definition K has a computable characteristic function, f. Thus, f can determine whether input x will make the x^{th} program, $\varphi_x(x)$, converge. Suppose the a^{th} program is:

$$\varphi_a(x) = \begin{cases} 1 & \text{if } f(x) = 0 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Therefore if $f(x)$ is computable, then $\varphi_a(x)$ is p.c. Observe that if $f(a) = 0$, then $\varphi_a(a)$ halts and by definition of p.c., $a \in K$, but $f(a) = 0$ indicates $a \notin K$. Similarly, if $f(a) = 1$, then $\varphi_a(a)$ diverges and $a \notin K$ but $f(a) = 1$ indicates $a \in K$. This is a contradiction and so, K is not a computable set. □

Thus there is no computable function that can decide whether any given computer program will halt or run on forever.

3. Post's Theorem: Jumps and Quantifiers

3.1. **Arithmetical Hierarchy.** We will now look at $\sum_n^0$, $\prod_n^0$, and Δ_n^0 sets (or $\sum_n$, $\prod_n$, and Δ_n for short) in which the n subscript signifies the number of quantifiers used for a computable relation. These sets are part of a concept called *Kleene arithmetical hierarchy* which was discussed in Kleene's paper, *Recursive predicates and quantifiers*, and book, *Introduction to Metamathematics*, written in 1943 and 1952 respectively [2] [1].

Definition 3.1. (i) A set B is contained in $\sum_0, \prod_0,$ or Δ_0 iff B is computable.
(ii) For $n \geq 1$, $B \in \sum_n$ if there exists a computable relation $R(x, y_1, y_2, ..., y_n)$ s.t:

$$x \in B \iff (\exists y_1)(\forall y_2)(\exists y_3)...(Qy_n)R(x, y_1, y_2, ..., y_n)$$

where Q is $\exists$ when n is odd and $\forall$ when n is even.
(iii) Likewise, $B \in \prod_n$ if

$$x \in B \iff (\forall y_1)(\exists y_2)(\forall y_3)...(Qy_n)R(x, y_1, y_2, ..., y_n)$$

where Q is $\forall$ when n is odd and $\exists$ when n is even.
(iv) $B \in \Delta_n$ if $B \in \prod_n$ and $B \in \sum_n$.

Lemma 3.2. *Given a set A and its complement, $\overline{A}$:*

$$A \in \sum_n \iff \overline{A} \in \prod_n.$$

Proof. $A \in \sum_n \iff (\exists y_1)(\forall y_2)(\exists y_3)...R(x, y_1, y_2, ..., y_n)$.
Thus, $\overline{A} = (\forall y_1)(\exists y_2)(\forall y_3)...\neg R(x, y_1, y_2, ..., y_n)$.
Which by definition means $\overline{A} \in \prod_n$.

□

3.2. **Degrees and Jumps.** In 1939, Turing published his PhD dissertation under Church, in which he discusses the *oracle Turing machine* [8]. This mechanism is an improvement upon the original Turing machine by adding to the structure an additional unmodifiable tape called the *oracle tape*, with its own reading head, the *oracle head*, to move at the same time as the original reading head. We will now consider the class of functions given by the *oracle machine algorithms* which have access to external databases of information known as *oracles*.

Definition 3.3. An *oracle* is a partial function, $g : \mathbb{N} \to \{0, 1\}$, that will provide the value of $g(n)$ whenever requested by the algorithm in the computation process, where n is some input determined computably by the machine running the algorithm. Just like for the original Turing machines, a list of possible algorithms can also be obtained from oracle machines. For the e^{th} oracle machine algorithm, Φ, with oracle, A, the function would be denoted by: Φ_e^A. Note that an oracle machine can compute every computable function with the oracle set as the empty set ($\emptyset$).

Turing's discussion of the oracle machine went unnoticed by mathematicians for years until Post utilized the notion of oracles to examine *Turing reducibility* in his groundbreaking 1944 paper, *Recursively enumerable sets of positive integers and their decision problems* [4]. Post's research is fundamental in computability theory since this field focuses more on comparing noncomputability rather than computability of sets which the concept of Turing reducibility allows for [5].

Definition 3.4. (i) A set A is *many-one reducible* (or *m-reducible*) to another set B ($A \leq_m B$) if there exists a computable function f such that $f(A) \subseteq B$ and $f(\overline{A}) \subseteq \overline{B}$, i.e $x \in A$ if and only if $f(x) \in B$.
(ii) A set A is *one-one reducible* (or *1-reducible*) to set B ($A \leq_1 B$) if $A \leq_m B$ and the computable function is 1:1.

Note that M-reducibility and 1-reducibility are stronger forms of *Turing reducibility*, which is the most general form of reducibility.

Definition 3.5. A set A is *Turing reducible* to another set B , ($A \leq_T B$), if the characteristic function of A, χ_A, can be computed by some oracle machine running on oracle B (χ_A is Φ_e^B for some e).

Post is the first to refer to this concept as Turing reducibility [5].

Definition 3.6. A set A is *Turing equivalent* to another set B, ($A \equiv_T B$), if A is Turing reducible to B and vice versa ($A \leq_T B$) and ($B \leq_T A$).

We will now look at an important concept of computability theory known as the *degree*. Turing degrees organize sets into levels of noncomputability in which all sets of the same degree have equal difficulty in determining their characteristic functions. The higher the degree of a set, the less computable it is.

Definition 3.7. (i) The *Turing degree* (also known as the *degree of unsolvability* or *degree* for short) of set A refers to the equivalence class of sets that are Turing equivalent to A.

$$\deg(A) = \{B : B \equiv_T A\}$$

where A and B are both sets.
(ii) A degree **a** is *computably enumerable* if it contains a c.e. set.
(iii) A degree **a** is *computably enumerable in* degree **b** (**a** is *c.e. in* **b**) if there exists some set $A \in \mathbf{a}$ that is c.e. in some set $B \in \mathbf{b}$.
(iv) A degree **a** is *computably enumerable in and above* degree **b** (**a** is *c.e.a in* **b**) if $\mathbf{a} \geq \mathbf{b}$ and **a** is c.e. in **b**.
Note that the concepts of being *c.e. in* and *c.e.a. in* apply to both sets and degrees.

Next we examine the concept of the *jump*. In essence, the jump increases the degree of the set, returning a set that is successively more difficult to determine its characteristic function.

Definition 3.8. (i) The *Turing jump* (or *jump* for short) of set A (denoted A' or K^A) is the halting problem relative to the oracle A, looking at whether $\Phi_x^A(x)$ halts.

$$A' = K^A = \{x : \Phi_x^A(x)\downarrow\} = \{x : x \in W_x^A\}.$$

(ii) The nth jump of set A, $A^{(n)}$, is the jump operator, the function that sends A to A', applied n times to A. Thus $A^{(0)} = A$ and $A^{(1)} = A'$.

Kleene and Post established many fundamental properties of both the jump and degree in their 1954 paper, *The upper semi-lattice of degrees of recursive unsolvability*, showing that $K^A >_T A$ and K^A is c.e. in A [3].

Theorem 3.9. *(Jump Theorem) Given sets A and B:*
(i) A' is c.e.a in A.
In essence, A' can't be computed by an oracle machine that uses an oracle for A.
(ii) $A \not\leq_T A$.
(iii) B is c.e. in $A \iff B \leq_1 A'$.
(iv) If A is c.e. in B and $B \leq_T C$ then A is c.e. in C.
(v) $B \leq_T A \iff B' \leq_1 A'$.
(vi) If $B \equiv_T A$ then $B' \equiv_1 A'$ (and therefore $B' \equiv_T A'$).
Thus if two sets have the same degree, then their jumps have the same degree.
(vii) A is c.e. in B iff A is c.e. in $\overline{B}$, the complement of B.

3.3. **The Zero-Jump, $\mathbf{0'}$.** The zero-jump, $\mathbf{0'}$, (pronounced zero-prime) refers to the equivalence class containing the jump of the empty set, $\emptyset$. Similarly, $\mathbf{0}^{(n)}$ refers to the degree of the nth jump of the empty set.

$$\mathbf{0}^{(n)} = \deg(\emptyset^{(n)}).$$

These degrees attribute important characteristics to the sets in its equivalence class. For instance the degree **0** contains all computable sets by definition:

$$\mathbf{0} = \deg(\emptyset) = \{B : B \text{ is computable}\},$$

while $\mathbf{0'}$ is the degree of the halting problem:

$$\mathbf{0} = \deg(\emptyset'), \text{ where } \emptyset' := K^{\emptyset} \equiv K.$$

The structure of these characteristics are further discussed in the next section.

3.4. **Post's Theorem.** Post's Theorem connects the concepts of arithmetical hierarchy and Turing jumps. Though Post did not publish this theorem with a proof, Kleene credits Post with the idea in his book, *Introduction to Metamathematics* [1][6].

Definition 3.10. (i) The *projection* of a relation, $R \subseteq \omega \times \omega$ is the set A such that:

$$A = \{x : (\exists y)R(x, y)\}.$$

(ii) A set A is $\sum_1^0$ if A is the projection of some computable relation, $R \subseteq \omega \times \omega$.
(iii) Similarly, a set A is in $\sum_1^{0,B}$*-form* (or A is $\sum_1^B$) if A is the projection of a relation computable with an oracle B (also known as a B-computable relation), $R^B \subseteq \omega \times \omega$.

Theorem 3.11. *(Quantifier Contraction Theorem) The Quantifier Contraction Theorem shows that several adjacent and identical quantifiers, whether universal or existential, can be simplified into one quantifier of the same type.*
(i) A set A is $\sum_1^0$ (i.e there exists a computable relation $R'(x, y)$ s.t $x \in A \iff (\exists y)R'(x, y)$) if and only if there exists $n \in \omega$ and a computable relation $R \subseteq \omega^{n+1}$ such that

$$A = \{x : (\exists y_1)...(\exists y_n)R(x, y_1, ..., y_n)\}.$$

(ii) Similarly, a set A is $\sum_1^B$ if and only if there exists $n \in \omega$ and a B-computable relation $R^B \subseteq \omega^{n+1}$ such that

$$A = \{x : (\exists y_1)...(\exists y_n)R^B(x, y_1, ..., y_n)\}.$$

(iii) Additionally, a set A is $\prod_1^0$ (i.e there exists a computable relation $R'(x, y)$ s.t $x \in A \iff (\forall y)R'(x, y)$) if and only if there exists $n \in \omega$ and a computable relation $R \subseteq \omega^{n+1}$ such that

$$A = \{x : (\forall y_1)...(\forall y_n)R(x, y_1, ..., y_n)\}.$$

(iv) Similar to (ii), set A is $\prod_1^B$ if and only if there exists $n \in \omega$ and a B-computable relation $R^B \subseteq \omega^{n+1}$ such that

$$A = \{x : (\forall y_1)...(\forall y_n)R^B(x, y_1, ..., y_n)\}.$$

Note that (ii) and (iv) are the quantifier contraction theorem relativised to oracle B.

Theorem 3.12. *A set B is c.e in another set A if and only if B is $\sum_1^A$.*

Definition 3.13. A set A is $\sum_n$*-complete* (or $\prod_n$*-complete*) if $A \in \sum_n$ (or $\prod_n$) and $B \leq_1 A$ for all $B \in \sum_n$ (or $\prod_n$).

Theorem 3.14. *(Post's Theorem). For all $n \geq 0$,*
(i) A set B is in $\sum_{n+1}$ if and only if the set B is computably enumerable in some $\prod_n$ set (which equivalently means it is computably enumerable in some $\sum_n$ set):

$$B \in \sum_{n+1} \iff B \text{ is c.e. in some } \prod_n \text{ set} \iff B \text{ is c.e. in some } \sum_n \text{ set.}$$

(ii) The n^{th} jump of the empty set, $\emptyset^{(n)}$, is $\sum_n$-complete for $n > 0$.
(iii) A set B is in $\sum_{n+1}$ if and only if the set B is computably enumerable in the n^{th} jump of the empty set:

$$B \in \sum_{n+1} \iff B \text{ is c.e. in } \emptyset^{(n)}.$$

(iv) A set B is in Δ_{n+1} if and only if the set B is Turing reducible to the n^{th} jump of the empty set:

$$B \in \Delta_{n+1} \iff B \leq_T \emptyset^{(n)}$$

.

Proof. (i)
"⇒"
$B \in \sum_{n+1} \Rightarrow$ there exists a computable relation $R(x, y_1, y_2, ..., y_n)$ s.t:

$$x \in B \iff (\exists y_1)(\forall y_2)(\exists y_3)...(Qy_n)R(x, y_1, y_2, ..., y_n)$$

$\Rightarrow$ by definition of projection (Def 3.10 (i)) set B is a projection of an A-computable relation R^A such that:

$$B = \{x : (\exists y)R^A(x, y)\}$$

for some set $A \in \prod_n$ by definition of $\prod_n$ (Def 3.1 (iii))
$\Rightarrow B$ is $\sum_1^A$ by relativised version of Quantifier Contraction Theorem (Thm 3.11 (ii))
$\Rightarrow B$ c.e. in $A \in \prod_n$ by Thm 3.12
$\therefore B$ c.e. in $\prod_n$
Note that $\overline{A} \in \sum_n$ by Lemma 3.2
$\therefore B$ c.e. in $\sum_n$ by Jump Theorem (Thm 3.9 (vii)).

"⇐"
Note the reverse direction of (i) can be proved by applying the above theorems in "⇒" in the reverse order since all these theorems are biconditional.

(ii) Proof by induction:
Step 1) Easy to show for $n = 1$ since this is essentially the halting problem (refer to section 2.
Step 2) Assume that for $n \geq 1$, $\emptyset^{(n)}$ is $\sum_n$ - complete.
Step 3) We now prove for $n + 1$:
Given set B where,
$B \in \sum_n \iff B$ is c.e. in some $\sum_n$ set by (i)
$\iff B$ c.e. in $\emptyset^{(n)}$ by induction
$\iff B \leq_1 \emptyset^{(n+1)}$ by Jump Theorem (Thm 3.9 (iii))
$\therefore \emptyset^{(n+1)}$ is $\sum_{n+1}$ - complete.
By induction, we are done and therefore $\emptyset^{(n)}$ is $\sum_n$ - complete.

(iii) By (ii), $\overline{\emptyset^{(n)}}$ is $\prod_n$-complete for $n \geq 1$.
(i) and (ii) imply (iii).

(iv) $B \in \Delta_{n+1} \iff B \in \prod_{n+1}$ and $B \in \sum_{n+1}$
$\iff B, \overline{B} \in \sum_n$ by Lemma 3.2
$\iff B$ and $\overline{B}$ are c.e. in $\emptyset^{(n)}$ by (iii)
$\iff B$ computable in $\emptyset^{(n)}$
$\iff B \leq_T \emptyset^{(n)}$ by definition (Def 3.5).
Post's Theorem can also apply to the jumps of non-empty computable sets since they would have the same degree. □

Again, it is important to note the strong link between the arithmetical hierarchy and computability established by Post's theorem. Such findings set the foundation for the study of Turing degrees structures. Research on the complex structure of degrees continues on, while many recent discoveries pertaining to this topic have

been made. In fact, computability theory as an entirety continues to be an actively researched field today [6].

Acknowledgments. It is a pleasure to thank my mentor, Jonathan Stephenson for all his guidance. I truly appreciate the invaluable insight and advice he provided during the research and writing process of this paper. I would also like to thank Peter May for his dedication to organizing this REU.

References

[1] S. C. Kleene. Introduction to Metamathematics. www.jstor.org/stable/3609805.
[2] S. C. Kleene. Recursive predicates and quantifiers. www.ams.org/journals/tran/1943-053-01/S0002-9947-1943-0007371-8/S0002-9947-1943-0007371-8.pdf.
[3] S. C. Kleene and Emil L. Post. The upper semi-lattice of degrees of recursive unsolvability. www.jstor.org/stable/1969708
[4] Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. www.ams.org/journals/bull/1944-50-05/S0002-9904-1944-08111-1/S0002-9904-1944-08111-1.pdf.
[5] Robert I. Soare. The Art of Turing Computability: Theory and Applications. Springer-Verlag, to appear.
[6] Klaus Ambos-Spies and Peter A. Fejer. Degrees of Unsolvability. cs.simons-rock.edu/cmpt320/history_of_degrees.pdf
[7] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. cs.virginia.edu/ robins/Turing_Paper_1936.pdf
[8] A. M. Turing. Systems of logic based on ordinals. plms.oxfordjournals.org/content/s2-45/1/161.