Diagonalization, Self-Reference & Incomputability

http://www.cse.iitd.ac.in/˜sak

S. Arun-Kumar

Department of Computer Science and Engineering

I. I. T. Delhi, Hauz Khas, New Delhi 110 016.

August 4, 2011

Contents

About Adjectives

Definition 1.1

- *An adjective in a language is* **autological** *iff it describes itself or can be applied to itself i.e. the meaning of the word is also a property of the word.*
- *Every adjective in English which is* <u>*not*</u> *autological is said to be* **hetero-logical**.

Example 1.2

- *"poly-sylla-bic" is a word consisting of 3 syllables, so it describes itself. Hence it is autological.*
- *"mono-sylla-bic" consists of more than 1 syllable, so it does not describe itself. Hence it is heterological.*

Examples

Autological	*Heterological*
adjectival	*adverbial*
single	*multiple*
polysyllabic	*monosyllabic*
English	*French*
olde	
unambiguous	*ambiguous*
man-made	

For any adjective "*adj*" ask the question

Is the word "adj" a(n) adj word?

If the answer is yes then it is autological, otherwise it is heterological.

Autological?

Is the word "autological" autological or heterological?

Autological?

Is the word "autological" autological or heterological?

- Suppose "*autological*" is an autological word. Then it describes itself. Therefore it *is* autological.

Autological?

Is the word "autological" autological or heterological?

- Suppose "*autological*" is an autological word. Then it describes itself. Therefore it *is* autological.
- Suppose "*autological*" is not an autological word. Then it does not describe itself. Therefore it *must* be heterological.

Autological?

Is the word "autological" autological or heterological?

- Suppose "*autological*" is an autological word. Then it describes itself. Therefore it *is* autological.
- Suppose "*autological*" is not an autological word. Then it does not describe itself. Therefore it *must* be heterological.
- *But then can it be both autological and heterological?*

autological $\oplus$ heterological

Heterological?

Is the word “heterological” autological or heterological?

Heterological?

Is the word "heterological" autological or heterological?

- Suppose "*heterological*" is an autological word. Then it describes itself. Therefore "*heterological*" must be a heterological word. But if it is heterological then it cannot be autological.

Heterological?

Is the word "heterological" autological or heterological?

- Suppose "*heterological*" is an autological word. Then it describes itself. Therefore "*heterological*" must be a heterological word. But if it is heterological then it cannot be autological.
- On the other hand if "*heterological*" is a heterological word, then it obviously *does* describe itself and hence by definition it must be autological. But then it cannot be heterological.

Heterological?

Is the word "heterological" autological or heterological?

- Suppose "*heterological*" is an autological word. Then it describes itself. Therefore "*heterological*" must be a heterological word. But if it is heterological then it cannot be autological.
- On the other hand if "*heterological*" is a heterological word, then it obviously *does* describe itself and hence by definition it must be autological. But then it cannot be heterological.
- *But "heterological" is an adjective. So it must be either autological or heterological!*

autological $\oplus$ heterological

Countability

Definition 2.1 *An infinite set is* **countable** *or* **countably infinite** *if it can be placed in 1-1 correspondence with* $\mathbb{N}$*. otherwise it is* **uncountable** *or* **uncountably infinite** .

Theorem 2.2 *The sets* $\mathbb{N}$ *and* $\mathbb{Z}$ *are countably infinite.*

■

Theorem 2.3 Prove it!

1. *The set* $\mathbb{N}^2$ *is countably infinite.*
2. *The set* $\mathbb{N}^n$ *for any* $n \geq 0$ *is countably infinite.*
3. *The set* $\mathbb{N}^* = \bigcup_{n \geq 0} \mathbb{N}^n$ *is also countably infinite.*
4. *For any (finite or) infinite set of arbitrary symbols, the set of all finite length sequences that can be formed is countably infinite.*

■

Uncountability

Theorem 2.4 **(The Powerset theorem).** *There is no 1-1 correspondence between a set and its powerset.*

Proof

Theorem 2.5 *Prove it!*

1. $2^{\mathbb{N}}$ *the powerset of the naturals is uncountably infinite.*
2. *The number of unary boolean functions* $b : \mathbb{N} \to \{0, 1\}$ *is uncountably infinite.*
3. *The number of unary functions* $f : \mathbb{N} \to \mathbb{N}$ *is at least uncountably infinite.*

■

Programs and Functions

Consider programs implementing unary functions on $\mathbb{N}$

- There are only *countably* many programs that can be written
- There are *uncountably* many unary functions that exist.
- Hence there are unary functions which cannot be programmed in any programming language. These functions are called **incomputable** functions.
- The functions for which programs can be written are the **computable** functions.

Proof of Powerset Theorem

Proof of theorem 2.4

Proof: Let A be any set and let $\mathbf{2}^A$ be its powerset. Assume that $g : A \to \mathbf{2}^A$ is a 1-1 correspondence between A and $\mathbf{2}^A$. This implies for every $a \in A$, $g(a) \subseteq A$ is uniquely determined and further for each $B \subseteq A$, $g^{-1}(B)$ exists and is uniquely determined.

For any $a \in A$, a is called an *interior* member if $a \in g(a)$ and otherwise a is an *exterior* member. Consider the set

$$X = \{x \in A \mid x \notin g(x)\}$$

which consists of exactly the exterior members of A. Since g is a 1-1 correspondence, there exists a unique $x \in A$ such that $X = g(x)$.

x is either an interior member or an exterior member. If x is an interior member then $x \in g(x) = X$ which contradicts the assumption that X contains only exterior members. If x is an exterior member then $x \notin g(x) = X$. But then since x is an exterior member $x \in X$, which is a contradiction. Hence the assumption that there exists a 1-1 correspondence g between A and $\mathbf{2}^A$ must be false.

Universal Machines

In a digital computer all programs and data (input and output) are represented as sequences of bits.

Digital computers with infinite memory are **universal machines**.

That is,
Given a digital computer with infinite memory one can write programs

- to simulate the working of any other digital computer with finite or infinite memory
- to simulate many discrete and continuous natural processes upto some approximation.

Simulation

Example 3.1 *If there is an integer adding machine AM such that*

$$AM(x, y) = x + y$$

then the working of this adding machine can be simulated by a program P_{AM} on a universal machine.

$$P_{AM}(x, y) = AM(x, y) = x + y$$

In general, given a universal machine UM, it is possible to write a program P_{UM} which for every (unary) function f that UM is capable of computing will yield

$$P_{UM}(f, x) = f(x)$$

Compilers & Interpreters

- Universality makes it possible to write *compilers* and *interpreters*.
- For any program P_L written in a languge L which takes an input x, the machine takes the language L, the program P_L and the input x and executes it.
- The universal machine essentially simulates a machine for P_L.

Universal Functions

1. There are only a countably infinite different programs that can be written in any programming languge L.
2. The set of all programs in any language L (that take a single natural number as input) can be enumerated by an algorithm.
3. Each program in the above enumeration implements a unary function on $\mathbb{N}$.
4. The set of all computable unary functions on $\mathbb{N}$ can be enumerated (since they are at most countably infinite).

$$h_0, h_1, h_2, \ldots \tag{1}$$

5. Some of the functions in the enumeration could be undefined on some or all of $\mathbb{N}$.

Definition 3.2 *A binary function $u : \mathbb{N}^2 \to \mathbb{N}$ is* **universal** *for all computable unary functions on $\mathbb{N}$, if for all $(x, y) \in \mathbb{N}^2$, $u(x, y) = h_x(y)$.*

Undefinedness

Let

$$h_0, h_1, h_2, \ldots \tag{2}$$

be an enumeration of all the (unary) computable functions on $\mathbb{N}$.

Theorem 3.3 *There is no computable function which will determine whether* $h_i(i)$ *is defined.*

Proof: The characteristic function for this problem is

$$f(x) = \begin{cases} 1 & \text{if } h_x(x) \in \mathbb{N} \\ 0 & \text{if } h_x(x) \notin \mathbb{N} \end{cases}$$

Suppose f is computable. Let

$$g(x) = \begin{cases} 0 & \text{if } f(x) = 0 \\ \perp & \text{if } f(x) = 1 \end{cases}$$

Since f is computable, so is g. Since g is a unary computable function on $\mathbb{N}$, g must occur in the sequence (2). Suppose $g = h_m$. Then $g(m) = 0 \in \mathbb{N}$ iff $f(m) = 0$ iff $h_m(m) \notin \mathbb{N}$ iff $g(m) \notin \mathbb{N}$ which is a contradiction. Hence the assumption that f is computable must be false. ■

Totality of Functions

Definition 3.4 *A function* $f : A \to B$ *is* **total** *if it is defined for every* $a \in A$*.*

Theorem 3.5 *There is no computable function which can determine whether any unary computable function is total.*

Proof: Let

$$g(x) = \begin{cases} 1 & \text{if } h_x \text{ is total} \\ 0 & \text{if } h_x \text{ is not total} \end{cases}$$

Assume g is computable. Let

$$\begin{aligned} f(x) &= \begin{cases} h_x(x)+1 & \text{if } h_x \text{ is total} \\ 0 & \text{if } h_x \text{ is not total} \end{cases} \\ &= \begin{cases} u(x,x)+1 & \text{if } g(x)=1 \\ 0 & \text{if } g(x)=0 \end{cases} \qquad (*) \end{aligned}$$

Since u is computable and g is computable, f must be computable. But f is total and different from every function in the sequence (2) since for each $x \in \mathbb{N}$, $f(x) \neq h_x(x)$. Hence f is not computable, which is a contradiction. ■

Computers and Unsolvability

1. Programmers often try to solve unsolvable problems.
2. A problem is unsolvable if there is no algorithm which solves the problem in *finite* time even with *unbounded* resources.
3. There are fundamental limitations of computers as we know them.

The Problem of Incomputability

- To prove a problem *can* be solved one is required to write a program (or *algorithm*) to solve the problem in a *(pseudo-)*programming language and *prove* that it works.
- But to prove that a problem cannot be solved requires a more indirect method.
- We have shown that there are *unsolvable* problems using a *diagonalization* and proof by contradiction.

Exercises

1. Prove all the parts of therorem 2.3
2. Prove all the parts of therorem 2.5

Thank You!

Any Questions?