

On the Concept of Enumerability Of Infinite Sets

Charlie Obimbo
Dept. of Computing and Information Science
University of Guelph

ABSTRACT





This paper discusses the concept of enumerability of infinite sets. This concept is applied in a number of areas, including Data Models & Query languages, Hypothetical Reasoning with Intuitionistic Logic, and Sequential Query languages. However, the concept is lost on most students, due to the fact that the Concept of enumerability is lost on them.

With the introduction of a new way to look at cardinality and enumerability of infinite sets, the students have found this a better way of looking at infinite sets, and have easily been able to apply it to the various areas needed.

Keywords: sets, cardinality, enumerable, infinite sets, languages.

1. Introduction

The concept of *infinitely enumerable sets* is used widely in many areas of Computer Science. Some of these areas include:

-  Data Models & Query languages [1]
-  Hypothetical Reasoning with Intuitionistic Logic [2]
-  Sequential Query Languages [3,4]
-  Proof of the existence of noncomputable problems like the Halting Problem [5,6]

to name but a few.

The understanding of this concept is convoluted due to the abstractness, and yet imprecise definitions given of cardinality of sets.

The concept used in most Discrete Mathematics books in introducing the concept of enumerability (which in some cases is erroneously called

countability) of infinite sets is that of set-cardinality and one-to-one correspondences.

The texts normally *imply* that two sets are of equal cardinality if the number of elements in each are the same. This statement is true for finite sets. For infinite sets, however, this premise does not hold --- but since not clearly stated, it leaves the feeling, to most students and readers, of it being a concept to be believed without thorough validation.

This Paper describes how this concept has been taught in the Department of Computing Science, at the University of Guelph, in the CIS4600 Course (Theory of Computation), and how it has enabled the Students to easily apply it in the various fields of application.

2. Overview

A brief description of why the concept of *infinite enumerability* evades most students and readers, and why it is important to understand them, including examples of application.

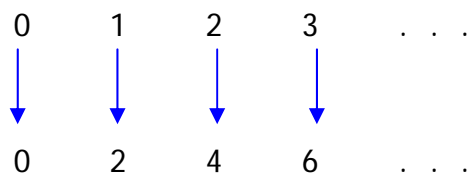
Also pitfalls in using basic concept of cardinality, as defined uniformly for finite and infinite sets.

Cardinality of a set is said to be the number of elements in the set.

Two sets S_1 and S_2 are said to be of the same cardinality, if there is a relation that maps every element of the set S_1 to the set S_2 and vice-versa. Using the cardinality notation, $|S_1| = |S_2|$.

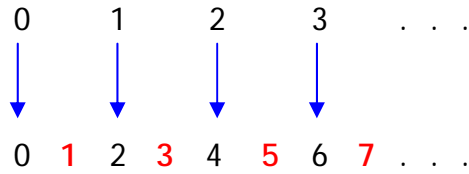
This definition works well for finite sets, but is rather confusing for infinite sets.

Example 2.1: The set of positive even numbers E is said to have the same cardinality as the set of Natural numbers N .



And yet, when we consider the two sets E and N , we find that $N - E$, that all the elements that are in N and that are not in E is the set of odd numbers O , which also happens to have the same cardinality as the set N Natural numbers.

The set O is depicted in the Figure below by the red numbers on the lower row.



Now this tends to depict that though the two sets are said to be of the same cardinality, they cannot not be said to have the same number of elements, and therefore they are not of the same size. Although we can say that they are of the same *relative-size*.

2.1 Sets and Enumerability

Let us consider the standard definitions of sets, subsets, and bijection (1-1 and Onto) mappings.

A set is a collection of objects. A set A is said to be a *subset* of a set B ($A \subseteq B$) if all elements in A are also in B . It is said to be a *proper subset* of B ($A \subset B$) if it is a subset of B and there is at least one element in B that is not in A .

In discussing functions, we talk about two sets: the *Domain* and the *Range*. If A and B are sets, then f is a function from A to B ($f: A \rightarrow B$) if for every element x in A , there is an element y in B , such that f maps x onto y . A is said to be the *Domain* of the function, and B the *Range*.

A function (a mapping) ($f: A \rightarrow B$) is said to be *one-to-one* (1-1) if f maps every element x in A to a unique element y in B (i.e. no two x 's share the same value y). Thus for $f: R \rightarrow R$ ¹ $f(x) = x^2$ is not 1-1, since $(-1)^2 = 1^2 = 1$, whereas $f(x) = x + 1$ is 1-1.

A function (or mapping) $f: A \rightarrow B$ is *onto* if every element of B is mapped onto by some element in A .

Thus if $f: N \rightarrow N$ [$N = \{0, 1, 2, \dots\}$] then $f(y) = 2x + 1$ is not onto, since $4 \in N$ is not mapped onto by any $x \in N$. However, for the same function, on the domain and range: $f: R \rightarrow R$, f is onto.

A function is said to be bijective, if it is 1-1 and onto.

¹ Note that R here denotes the set of Real Numbers.

Figure 2.1 illustrates some examples of the different types of Functions.

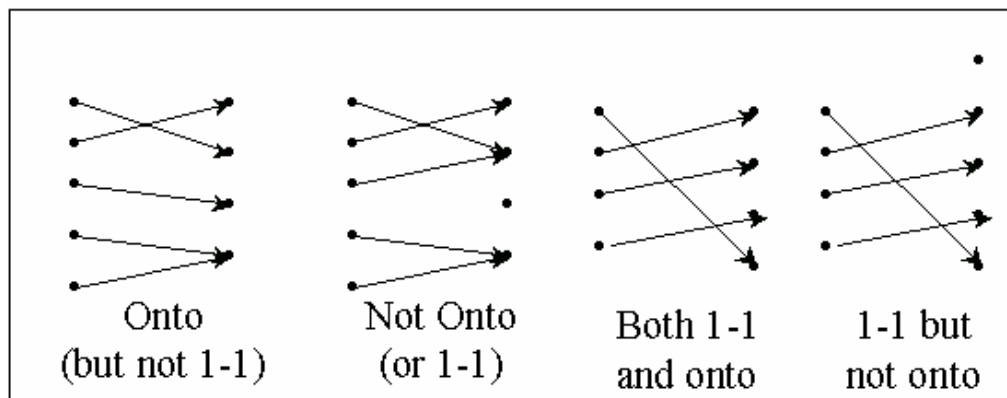


Figure 2.1

3. Enumerability of Sets

New Definition: A set, A , is said to be **enumerable** if there is a bijection between A and a subset of natural numbers N .

This definition applies both to finite and infinite sets. For finite sets, this would imply that, say if $M \subset N$, and $g: M \rightarrow A$ is a bijective function, then M has the same cardinality (and in fact the same size) as A .

On the other hand, a bijection between N and an infinite set B does not imply that they are the same size - but rather that there is an ordering - where indeed we can label the 1st element of B , the 2nd element etc.

An example, is enumerating valid programs in a programming language (say C). We start from a program with no lines of code (0) (if valid), then lexicographically all valid programs with 1 character, etc.

Indeed the relation, in this case, as in the case of $f: N \rightarrow E$ in Example 2.1 does imply a bijective function, but not that the two infinite sets are the same "size".

The other notion, about "counting", is that counting does imply a complete, or heading to completion term. In other words, a set is countable if we can "count" and finish "counting". And thus two sets are of the same size (cardinality) if we can count one, and the other, and see that the finished

number is the same. Indeed cardinality of infinite sets takes on a different meaning than this.

This new understanding, of infinite sets being *enumerated*, as opposed to being "*counted*" gives credibility and logic to the subject matter. We are just ordering, and not counting.

4. Application

One of the areas of application of this concept is in the Proof of the existence of noncomputable problems like the Halting Problem [5,6]. Below we give a brief description of the Halting Problem and outline its proof. This proof was demonstrated to the CIS4600: Theory of Computation class, at the University of Guelph, with much success after an understanding of the concept of enumerability.

The **halting problem** can be described as follows: It asks whether there is a procedure that takes as input, a computer program, and input to the program, and determines whether the program will eventually stop when run with this input. This would be a good program to have for **Software Verification** purposes.

4.1 Definition of the Halting Problem

- **Input:** Turing Machine M & Input x
- **Question:** Does Turing Machine M halt on input x ?

4.2 Language Theory Version of the Halting Problem

- We use M to represent both the abstract Turing Machine M and its string representation. Likewise, we use x to represent both the abstract input x and its string representation.
- $H = \{M; x \mid M \text{ halts when given input } x\}$
- Question: Is y in H ?

4.3 Theorem: H is not recursive

Proof: Diagonalization Proof [9]

1. Each Turing Machine is represented by some string over our alphabet.
2. Let E be an enumeration of all these strings (and thus all Turing Machines).
3. Assume that H is recursive.
4. This implies that there exists a TM M_H which decides H .

5. Use TM MH and enumeration E to construct a Turing Machine D which differs from each Turing Machine in E.
6. This creates a contradiction as E is an enumeration of all Turing Machines
7. Thus, our assumption in step 3 must be incorrect.
8. Thus, we conclude that H is not recursive.

5. Student's Reaction and Conclusions

This concept of enumerability of infinite sets, as opposed to countability, was taught to the CIS4600: Theory of Computation class, at the University of Guelph. The students were able to have a much better appreciation to cardinality of infinite sets, and their enumerability. This was demonstrated in their class participation, and their ability to apply it both to abstract and practical assignment problems.

6. References

1. Jan Paredaens & Bart Kuijpers. Data Models and Query Languages for Spatial Databases. University of Antwerp.
2. D. Miller. Lexical scoping as universal quantification. In G. Levi and M. Martelli, editors, Logic Programming: Proceedings of the Sixth International Conference, pages 268--283, Cambridge, MA, 1989. MIT Press.
3. J. Ullman: "Principles of database and knowledge-base systems, Vol. I", Computer Science Press, 1988.
4. Jan Van den Bussche. A Formal Basis for Extending SQL to Object-Oriented Databases. Bulletin of the European Association for Theoretical Computer Science, vol 40, pp 207- 216, 1990.
5. John Martin. Introduction to Languages and the Theory of Computation. 3rd Ed. McGraw Hill, 2003.
6. Michael Sipser. Introduction to the theory of Computation. PWS Publishing Company, 1996.
7. Kenneth Rosen. Discrete Mathematics and Its Applications. McGraw-Hill, September 2002.
8. Susanna S. Epp. Discrete Mathematics with Applications 2nd Ed. PWS Publishing Co., September 1996.
9. Notes: Eric Torng, Associate Professor in the Department of Computer Science at Michigan State University.



Author

Charlie F. Obimbo

Dept. of Computing and Information Science

University of Guelph

Guelph, ON N1G 2W1

cobimbo@cis.uoguelph.ca

<http://www.cis.uoguelph.ca/~cobimbo>